LA RECURSIÓN EN LA CIENCIA COGNITIVA DEL LENGUAJE: UNA INVESTIGACIÓN FORMAL Y EXPERIMENTAL

TESIS DOCTORAL

Sergio Mota

UNIVERSIDAD AUTÓNOMA DE MADRID
PROGRAMA DE DOCTORADO EN
PSICOLOGÍA CLÍNICA Y DE LA SALUD
FACULTAD DE PSICOLOGÍA
2016

LA RECURSIÓN EN LA CIENCIA COGNITIVA DEL LENGUAJE: UNA INVESTIGACIÓN FORMAL Y EXPERIMENTAL

TESIS DOCTORAL

AUTOR
Sergio Mota Verdura

DIRECTOR
José Manuel Igoa González

UNIVERSIDAD AUTÓNOMA DE MADRID
PROGRAMA DE DOCTORADO EN
PSICOLOGÍA CLÍNICA Y DE LA SALUD
FACULTAD DE PSICOLOGÍA
2016

«*Eine ganze Wolke von Philosophie kondensiert zu einem Tröpfchen Sprachlehre*».
[«Toda una nube de filosofía se condensa en una gotita de gramática».]

Wittgenstein, *Investigaciones Filosóficas*, 1988, XI, p. 507.

«La confusión y esterilidad de la psicología no se puede explicar por el hecho de que es una «ciencia joven»; no se puede comparar su estado, por ejemplo, con el de la física en sus comienzos. (En todo caso más bien con el de ciertas ramas de la matemática. Teoría de conjuntos.) En efecto, en psicología existen métodos experimentales y *confusión conceptual*».

Wittgenstein, *Investigaciones Filosóficas*, 1988, XIV, p. 525.

«La presencia del método experimental nos hace creer que ya disponemos de los medios para librarnos de los problemas que nos inquietan; cuando en realidad problemas y métodos pasan de largo sin encontrarse».

Wittgenstein, *Investigaciones Filosóficas*, 1988, XIV, p. 527.

A mis padres y a mi hermano

Índice

PREFACIO

Este trabajo es el resultado de varios años de reflexión sobre un tema que me ha llevado por varias disciplinas y saberes, algunos de los cuales figuran curricularmente, mientras que otros reflejan mi forma de pensar, que se muestra en lo que sigue a estas líneas. A mi entender, el tema sobre el que escribo dejó hace tiempo de ser importante en sí mismo, por lo que en este trabajo se presenta más bien como un pretexto para un asunto de mayor calado, el cual tiene que ver con la caracterización de ciertos problemas en psicología como *problemas* y de ciertas prácticas consideradas como su *solución*.

En este sentido, lo que aquí se ha logrado, si es que se ha logrado algo, no tiene mayor valor para otros del que tiene para mí. Para bien o para mal, este trabajo no le rinde cuentas a ninguna subvención. Ninguna subvención ha delimitado temporalmente ni constreñido institucionalmente este trabajo. Los límites del mismo han sido establecidos bien temática o bien *gramaticalmente*.[1]

Además, esta tesis ha de entenderse, y de hecho no hay otra forma cabal de entenderla, como un punto, un vértice, en el que confluyen diferentes áreas de conocimiento, sus aristas, y como tal, es sólo un vértice de la representación *gramatical* perspicua a la que la Psicología debería aspirar. Según se irá haciendo evidente, esta tesis sólo recoge, siquiera levemente, aquello que tiene que ver con una parte mínima del estudio del lenguaje. Lo más importante de ella no son las conclusiones más o menos generales, más o menos precisas, sino la posibilidad de aplicación de un método a la clarificación de ciertos *problemas*. Este método muestra, entre otras cosas, que no todos los *problemas* en Psicología han de resolverse experimentalmente. Y ello no es porque tales problemas queden fuera de la Psicología, al contrario, están en la esencia de la misma, en su gramática.

S.M.V. Madrid, 2015

[1] En esta Introducción daré buena cuenta del sentido particular, no lingüístico, en el que emplearé el adjetivo "gramatical" (o al adverbio derivado "gramaticalmente"). De ahí su escritura en letra cursiva.

Agradecimientos

No me corresponde a mí juzgar este trabajo, lo cual no quiere decir que no tenga una idea general del mismo. Pero sí puedo decir que ha tenido un importante efecto en mí. Esto quiere decir que ha tenido importantes consecuencias sobre la propia comprensión, sobre la propia manera de ver las cosas. Esto significa que si este trabajo pudiera ofrecer algo a otros lectores, eso pasaría necesariamente porque previamente ha afectado a mi propia manera de ver las cosas. Por supuesto, estoy seguro de que este trabajo es mejorable, no me cabe duda alguna, y por eso, quiero mostrar mi agradecimiento a diferentes personas que han pretendido mejorar este trabajo de una manera u otra. Mi gratitud a José Hierro y a Anastasio Alemán por su disponibilidad, interés y calidad tanto humana, como intelectual. También deseo expresar mi agradecimiento a diferentes autores que se han tomado el interés y la molestia de leer mis manuscritos, o algunos de ellos, y que han aportado valiosos comentarios, como Noam Chomsky, Luis Eguren, David J. Lobina, Carlos U. Moulines, Jorge Ruiz Abánades, Roberto Torretti y Vicente Sanfélix.

Por último, toda mi gratitud a José Manuel Igoa, por su sabiduría, no sólo en relación con el aspecto académico de este trabajo, sino, y quizá más importante, por todas las conversaciones sobre la gramática de *lo académico*, sobre cómo las personas constituyen este mundo y sobre cómo hablan de él. Sobre lo que se dice y lo que se muestra. Sus límites a mi trabajo han sido mis límites, me ha cedido el espacio y el tiempo necesarios, no ha habido constricción ni objeción alguna en la elección de mis interlocutores y esto, más allá de cualquier muestra de ánimo e incluso más allá de cualquier ayuda explícita –que obviamente también agradezco–, es lo más importante. Ha sabido ver y entender lo que para mí ha significado esta tesis doctoral.

1. Introducción general

1.1. Sobre el concepto de recursión

En la tesis doctoral que tenemos entre manos me he propuesto analizar el uso de la noción de recursión en la Ciencia Cognitiva del lenguaje, desde una perspectiva tanto formal (i.e., conceptual o gramatical) como experimental (i.e., empírica). Este es el tema, podemos decir, general que subyace a todos los artículos que la componen. En aras de alcanzar este objetivo, haré un recorrido por diversos dominios de conocimiento, algunos de los cuales caen fuera del ámbito de la ciencia cognitiva propiamente dicha. Al decir "caen fuera" no pretendo sugerir que no tengan nada que ver con ella. Esta aclaración es debida a la reiterada opinión de revisores anónimos que insisten en que ya es hora de que nos centremos en la ciencia cognitiva y dejemos de mirar fuera de ella para analizar el uso de este término. Sin embargo, esta mirada al exterior resulta imprescindible, ya que el concepto de recursión tiene su origen precisamente *fuera* de la ciencia cognitiva.

Con todo, antes de definir el concepto de recursión en sus términos originales, es interesante resaltar, aunque sea brevemente, que dicha noción ha despertado un enorme interés dentro de la ciencia cognitiva del lenguaje, sobre todo desde que en un artículo ya clásico, Hauser, Chomsky y Fitch (2002) situaran esta propiedad en el centro mismo de la facultad humana del lenguaje. Como no podía ser de otro modo, no todos los autores han estado de acuerdo con esa formulación y por ello, la propuesta radical de Hauser, Chomsky y Fitch ha suscitado interesantes debates con otros autores, entre los que destacan Pinker y Jackendoff (2005; véase también Jackendoff y Pinker, 2005 y la correspondiente réplica de Fitch, Hauser y Chomsky, 2005). En términos generales, la discusión gira en torno a si la propiedad de la recursión es un rasgo específico del lenguaje, específico de la especie humana o ambas cosas. Autores como Jackendoff (2009), Katz y Pesetsky (2009), por un lado, o Corballis (2007, 2011), por otro, han propuesto que la recursión podría ser también una propiedad de la facultad musical o una propiedad del pensamiento, respectivamente. Sin embargo, lo que estos y

otros autores han entendido por ‘recursión’ no es lo mismo en todos los casos. Así, Jackendoff se ha centrado en las estructuras jerárquicas musicales, y en concreto en la propiedad de la auto-inclusión, que presuntamente las caracteriza. Así, este autor sostiene que las estructuras musicales son recursivas porque cada constituyente de una pieza o frase musical, dominado por un núcleo, puede, a su vez, contener otro núcleo. Sin embargo, al referirse a la recursión en estos términos, Jackendoff está aludiendo a la propiedad de la auto-inclusión, y no a la recursión *sensu stricto*. Katz y Pesetsky (2009), por su parte, sí han tratado de definir recursivamente un mecanismo computacional semejante al procedimiento *Merge* de la lingüística generativa (Chomsky, 1995), pero aplicado a las estructuras musicales. No obstante, ambos intentos, es decir, la definición de la recursión en las estructuras lingüísticas y en las estructuras musicales, son empresas muy diferentes. Así, como Chomsky (2015, p. 94) ha señalado, la recursión se ha confundido muy a menudo con la inclusión central (*center-embedding*), cuando en realidad se trata de dos nociones muy diferentes. De esto se sigue que la auto-inclusión (*self-embedding*), una noción más estrecha que la anterior (Chomsky, ibid., p. 95), pues aplicada sobre tales estructuras exige que el constituyente que está incluido en otro sea del mismo tipo, también es una noción muy distinta de la de recursión, lo que contribuye a confundir aún más el debate.

Por otro lado, autores como Kinsella (2010) han señalado la posibilidad de que la recursión sea una propiedad de dominios ajenos al lenguaje, así como una característica compartida con otras especies. En esta línea, Gentner et al. (2006) han tratado de demostrar empíricamente que la propiedad de la recursión no es exclusiva de la especie humana, basándose en la observación de que, por ejemplo, los estorninos, una especie de aves canoras, son capaces de aprender una gramática del tipo A^nB^n –esto es, una gramática definida como recursiva- después de un entrenamiento apropiado. Una gramática tal genera secuencias del tipo AABB (o AAABBB, AAAABBBB, etc.). A este respecto, los siguientes dos comentarios son importantes. Por una parte, una secuencia semejante puede o no ser recursiva (en términos de auto-inclusión); es decir, puede describirse, como

ya indicara Corballis (2007), alternativamente como una jerarquía de secuencias AB anidadas dentro de otras secuencias idénticas (como en [A[AB]B]), donde existe una dependencia entre unidades no adyacentes dispuestas jerárquicamente, o bien como una sucesión de unidades de un tipo (A) seguida del mismo número de unidades de otro tipo (B) (así, [AA][BB]), sin que haya tales dependencias. Por otra parte, en un análisis de esta naturaleza se confunde la generación y el procesamiento de tales estructuras con la organización interna de las mismas. A tenor de lo dicho anteriormente, una gramática libre de contexto como A^nB^n puede generar las secuencias antedichas mediante reglas no recursivas. Así:

Para $n = 3$, A^n = AAA; B^n = BBB. Resultado: [AAA][BBB]

Asimismo, y de forma complementaria, una gramática de estados finitos como $(AB)^n$, considerada no recursiva y que produce series como ABABAB, puede generar secuencias mediante el siguiente sistema de reglas, siendo una de ellas recursiva: S→ABS; S→AB.

Lo que estas observaciones pretenden poner de manifiesto es que una mejor aclaración del uso del término ‘recursión’ ayudará a clarificar todas estas controversias. En este sentido, considero oportuno distinguir entre la definición por recursión aplicada a la definición de un procedimiento generativo/computacional, como hace Chomsky en relación con el procedimiento denominado *Merge* en su Programa Minimista (Chomsky, 1995), y la noción de recursión como ‘auto-inclusión’, que se emplea comúnmente para caracterizar la organización interna de las estructuras. Esta distinción nos autoriza a preguntar para qué usar ‘recursión’ en este caso cuando ya tenemos una palabra para ello, a saber, ‘auto-inclusión’.

A tenor de las anteriores reflexiones, y antes de pasar a otros objetivos derivados del objetivo general enunciado al principio de esta introducción, permítame el lector ofrecer un excurso sobre la definición por recursión en la lógica matemática, su ámbito original de aplicación (cf. Soare, 1996).

Hablar sobre recursión en general o ‘en el vacío’ tiene poco sentido y, por ello, conviene aclarar qué quiero decir cuando hablo de recursión. En

primer lugar, uso este término en su sentido original para significar una regla, una definición, aplicada ampliamente en la lógica matemática para definir funciones o predicados (Kleene, 1952). Un ejemplo bien conocido lo podemos encontrar en la obra de Richard Dedekind (1888) o en la de Giuseppe Peano (reproducido en Mosterín, 2007), en donde podemos encontrar la definición recursiva de la suma:

$n+1 = n'$ [el caso base],[1]
$n+m' = (n+m)'$ [el paso recursivo].

Como suele ser habitual en la literatura (véase por ejemplo, Kleene, 1952; Cutland, 1980; Soare, 1996; Odifreddi, 2001) encontramos que una función recursiva es aquella que se define para un argumento x haciendo uso de sus propios valores previamente computados para argumentos menores (i.e., $f(y)$ para $y < x$). Asimismo, suelen emplearse funciones más simples y, generalmente, previamente definidas, como son la funciones iniciales *sucesor*, *constante* e *identidad*. Las funciones recursivas son obtenidas por medio de estas funciones iniciales a las que se aplica un número finito de veces los procesos de *definición por substitución* y de *definición por recursión*. A las funciones iniciales les corresponde los tres primeros esquemas (i.e., I-III) expuestos a continuación, mientras que a la definición por sustitución le corresponde el esquema IV y a la definición por recursión le corresponde el esquema V. Siguiendo a Kleene (1943, p. 42):

Funciones iniciales

(I) $\phi(x) = x'$
(II) $\phi(x_1, \ldots, x_n) = c$
(III) $\phi(x_1, \ldots, x_n) = x_i$

[1] En otras ocasiones, el sucesor también se expresa como $s(n)$. asimismo, en otros lugares el caso base se suele expresar como $a+0 = a$.

Esquema de definición por substitución

(IV) $\phi(x_1, \ldots, x_n) = \theta(\chi_1(x_1,\ldots, x_n), \ldots, \chi_m(x_1,\ldots, x_n))$.[2]

Esquema de definición por recursión[3]

(Va) $\phi(0) = c$,
$\phi(y') = \chi(y, \phi(y))$.

(Vb) $\phi(0, x_1, \ldots, x_n) = \psi(x_1, \ldots, x_n)$,
$\phi(y', x_1, \ldots, x_n) = \chi(y, \phi(y, x_1, \ldots, x_n), x_1, \ldots, x_n)$.[4]

Las funciones obtenidas mediante estos esquemas son denominadas funciones recursivas primitivas. Pero no son las únicas clases de funciones recursivas que se pueden definir. Así, Gödel (1934), definió las clases de las *funciones recursivas generales*, que incluye a las primitivas, y Kleene (1938, 1943, 1952) definió la clase de las *funciones recursivas parciales*, que incluye tanto a las generales como a las primitivas. Para definir tanto la clase de las funciones recursivas generales como la clase de las funciones recursivas parciales es suficiente con añadir, a los esquemas previamente definidos, un esquema más, a saber, el de *minimalización* (VI), que hace uso del operador μ (y que se lee "el mínimo…tal que"; cf. Kleene, 1943, p. 45):

(IV) $\phi(x_1, \ldots, x_n) = \mu y[\rho(x_1, \ldots, x_n, y) = 0]$.[5]

---

[2] ϕ está definida por substitución con ayuda de θ y χ si y sólo si para cualesquiera números naturales $x_1, \ldots, x_n$ ocurre que: $\phi(x_1, \ldots, x_n) = \theta(\chi_1(x_1,\ldots, x_n), \ldots, \chi_m(x_1,\ldots, x_n))$ (cf. Mosterín, 2007, p. 377).

[3] El esquema Va es un esquema de definición por recursión para una variable sin parámetros, mientras que el esquema Vb es un esquema de definición por recursión para una variable con parámetros. Así, Va y Vb son *versiones* del esquema de definición por recursión, pudiéndose ampliar a otras versiones en las que, por ejemplo, se hace uso de la definición por recursión para dos variables simultáneamente.

[4] ϕ está definida por recursión con ayuda de ψ y χ si y sólo si para cualesquiera números naturales $y, x_1 \ldots x_n$ ocurre que: $\phi(0, x_1, \ldots, x_n) = \psi(x_1, \ldots, x_n)$; $\phi(y', x_1, \ldots, x_n) = \chi(y, \phi(y, x_1, \ldots, x_n), x_1, \ldots, x_n)$ (cf. Mosterín, 2007, p. 377). En este caso, ejemplifico el esquema con Vb, pero sería extrapolable *mutatis mutandis* a todas las versiones del esquema V.

[5] Una función ϕ está definida por minimalización a partir de una función ρ en caso normal si y sólo si para cada $x_1, \ldots, x_n$ existe al menos un y tal que $\rho(x_1, \ldots, x_n, y) = 0$, y para cualesquiera números naturales $x_1, \ldots, x_n$ ocurre que: $\phi(x_1, \ldots, x_n) = \mu y[\rho(x_1, \ldots, x_n, y) = 0]$ (cf. Mosterín, 2007, p. 379).

Así las cosas, la clase de las funciones recursivas generales se cierra bajo I-VI, teniendo que cumplir la función definida mediante el esquema VI la exigencia de que tiene que tener al menos una solución.[6] Al relajar esta exigencia tenemos las funciones recursivas parciales, también definidas mediante I-VI, pero en este caso, una función definida mediante VI no tiene que tener necesariamente una solución (véase Torretti, 1998, para más detalles). Como puede apreciarse siguiendo a Kleene (1943), todas las *clases* de funciones recursivas hacen uso del esquema de definición por recursión (en alguna versión u otra, de hecho los esquemas I-V son *transversales* a todas las *clases*) lo cual no quiere decir que todos sus *miembros* hagan uso de dicho esquema.[7]

Dada la forma en que se definen las funciones recursivas, nos indican que son computables y de la propia definición se sigue la manera en que se han de ir computando sucesivamente sus valores (i.e., el modo en que dicho procedimiento es implementado; lo que se conoce como *proceso* recursivo). Así, para computar 2+4 (en donde '2' es el valor del parámetro fijo y '4' es el valor de la variable) tenemos el siguiente esquema de definición por recursión:

2+0 = 2 [caso base]
2+4 = (2+3)+1 [paso recursivo]

2+4 = (2+3)+1,
= ((2+2)+1)+1,
= (((2+1)+1)+1)+1,
= ((((2+0)+1)+1)+1)+1,
= 6

[6] Como señala Mosterín (2007, p. 379), la conocida función de Ackermann es una función recursiva (general); esto es, es una función definible a partir de las funciones iniciales (esquemas I-III) por medio de un número finito de aplicaciones de definiciones por sustitución (esquema IV), por recursión (esquema V), y por minimalización en caso normal (esquema VI). Un ejemplo de tal función es como sigue: $f(0, y) = S(y)$; $f(x+1, 0) = f(x, 1)$; $f(x+1, y+1) = f(x, f(x+1, y))$. La función que se define es f, con ayuda de S, la función sucesor.

[7] Mi gratitud al profesor Roberto Torretti por haberse tomado la molestia y el interés en atender a mis comentarios.

Como puede apreciarse en el ejemplo anterior, la definición recursiva "es una regla para la construcción de reglas de sustitución" (Wittgenstein, 1974, 36, p. 851).

Como ya he comentado al inicio de esta introducción, conviene diferenciar este uso del término 'recursión', que es el uso original y primario, de otro presuntamente relacionado con él, a saber, aquel que tiene que ver con la noción de auto-inclusión. Este es un objetivo más específico de esta tesis y derivado del objetivo general, dado que ambos sentidos del término aparecen relacionados en la literatura especializada, en la que es frecuente encontrar la afirmación de que una estructura recursiva es una estructura con auto-inclusión. En los artículos que siguen a esta introducción, el lector encontrará referencias bibliográficas sobre esta identificación tan común; permítaseme, por tanto, evitar repetir la extensa serie de citas. En el ejercicio de esta distinción, concluyo que no hay ninguna necesidad, ni teórica ni empírica, para hablar de recursión cuando hablamos de auto-inclusión. Para empezar, la recursión no sólo se predica de entidades teóricas distintas, funciones o procedimientos, en el primer caso, y estructuras de datos, en el segundo, sino que, además, cuando hablamos de recursión, en cada caso estamos hablando de propiedades diferentes. Así, cuando definimos recursivamente una función, estamos aplicando reglas, reglas de substitución, como ya he indicado, y no nos referimos a estructuras con auto-inclusión. En cambio, cuando hablamos de estructuras recursivas, hablamos de estructuras cuya organización interna se compone de constituyentes incluidos (o incrustados) dentro de constituyentes del mismo tipo (Wirth, 1986; Pinker y Jackendoff, 2005; por citar algunos). Un ejemplo servirá para ilustrar este punto. Wirth (1986, p. 110) indica que una expresión como 2+2 (para el caso y+z) exhibe una estructura recursiva como la siguiente:

+	
T	*y*
T	*z*

Esta expresión está encabezada por el operador '+', el cual relaciona dos variables, operandos o términos (T), '*y*' y '*z*'. En este sentido, un componente de la estructura (i.e., el componente de la estructura constituido por '+' y el término (T) '*y*') está contenido en otro componente del mismo tipo (i.e., el componente compuesto por el operador '+' y el término (T) '*z*').

Esto es así de manera independiente de la regla recursiva; es decir, un paso recursivo como '2+2 = (2+1)+1' no es recursivo porque una expresión que aparece en la misma exhiba una organización interna como la mostrada en el ejemplo. Si así fuera, '2+2 = 4' sería una regla recursiva, pero como puede apreciarse con facilidad, tal regla no es una manifestación directa de '$a+(b+1) = (a+b)+1$'.

Este ejemplo es análogo a lo que podemos encontrar en la ciencia cognitiva del lenguaje. Así, una cosa es que un procedimiento mecánico finito, como pueda ser el que Chomsky ha definido dentro del Programa Minimista (1995), genere expresiones con auto-inclusión y otra cosa es que tal procedimiento se defina recursivamente.[8] Empezando por esto último, dentro del actual *Programa Minimista*, Chomsky (cf. 1995, 2007) establece que un lenguaje-I es un procedimiento generativo/computacional que genera recursivamente una ordenación (o serie) infinita de expresiones jerárquicamente estructuradas mediante la operación denominada *Merge* (o 'ensamble'.[9] Lo que hace un procedimiento mecánico finito de este tipo es construir recursivamente objetos sintácticos (Chomsky, 1995, p. 226). Tales expresiones u objetos sintácticos generados por *Merge* son interpretados por dos sistemas, a saber: el sensorio-motor y el conceptual-intencional (cf. por ejemplo Hauser Chomsky y Fitch, 2002).

Una propiedad esencial (y a la vez elemental) del lenguaje humano que está directamente vinculada a su carácter generativo y que, por tanto, emana

[8] El lector encontrará un desarrollo más amplio de este tema en los artículos que siguen a esta introducción, fundamentalmente en los tres primeros.

[9] Para Chomsky (1959, 1995, 2005, 2007, 2008, 2010, 2011, 2015), el Lenguaje-I es concebido como un procedimiento generativo o sistema de cómputo capaz de generar una cantidad potencialmente infinita de expresiones jerárquicas internas. De este modo, la teoría de la gramática, que comprende el estudio del lenguaje así entendido, es el estudio de una clase especial de funciones recursivas dentro de la teoría de la computación (o computabilidad) (Chomsky, 1959, 2015).

inmediatamente del procedimiento computacional en el que consiste el lenguaje-I es su productividad infinita o, por usar la expresión al uso, su 'infinitud discreta' (Chomsky, 2000). Dada la asociación explícita que a menudo se establece entre recursión e infinitud discreta, considero de interés en este punto analizar la relación entre ambos conceptos. En primer lugar, la infinitud discreta tiene que ver con la posibilidad de generar, por medio de un conjunto finito de elementos, una ordenación potencialmente infinita de expresiones. En este sentido, y contra lo que señalan autores como Vicari y Adenzato (2014, p. 173), la infinitud discreta no se produce en virtud de la auto-inclusión. De hecho, Chomsky no se ha basado en la auto-inclusión para justificar la infinitud discreta. Antes bien, Chomsky (1956, p. 114) señaló expresamente que la recursión y la infinitud discreta no deberían confundirse, dado que las gramáticas de estado finito pueden también generar una infinitud discreta de elementos. En consonancia con ello, los mencionados Vicari y Adenzato (Ibid.) indican que hay sistemas con poder generativo infinito que no son recursivos, tales como los procedimientos iterativos. Y en efecto, *Merge* se puede implementar iterativamente y no por ello deja de tener poder generativo (esto se analiza en los artículos primero y segundo de esta tesis). Más aún, Chomsky ha señalado recientemente (2015, p. 94) que se asume incorrectamente que la recursión es necesariamente infinita. Al objeto de remarcar la independencia de la recursión y la infinitud discreta, en el segundo artículo de la tesis presento el argumento de que una función recursiva parcial puede dar un único valor o ninguno (puede estar definida sólo para un argumento o puede no estar definida para ninguno).

Veamos, pues, con más detalle, en qué consiste el procedimiento generativo *Merge*. Tal y como he defendido en diferentes lugares (Mota, 2013, 2014, 2015a, 2015b), la forma general de *Merge* se puede establecer de la siguiente manera: [N, O_s, $M(O_s)$]. N hace referencia a las piezas léxicas (o ítems léxicos) y objetos sintácticos que configuran la numeración (o repertorio seleccionado de objetos que entran en una derivación sintáctica), O_s hace referencia a un objeto sintáctico cualquiera de la serie generada (que puede tomar n-valores: si n = 1, entonces Os = X; si n = 2,

entonces *Os* = X, Y, y así sucesivamente) y $M(O_s)$ hace referencia a un objeto sintáctico nuevo generado a partir de O_s mediante la aplicación de *M* ( ) –esto es, *Merge*. Este procedimiento genera/define recursivamente objetos sintácticos, independientemente de cuál sea su organización interna a efectos de la auto-inclusión de constituyentes (o esquemas).

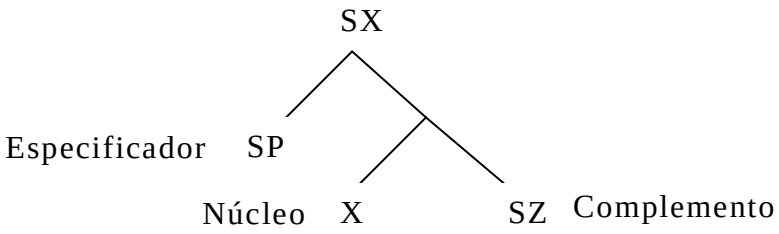
Lo que hace *Merge* (entendida aquí como una operación binaria) es tomar dos ítems léxicos, X,Y, para crear con esa unión un nuevo objeto sintáctico Z (X,Y = {X,Y} = Z). Mediante otra aplicación, *Merge* genera un nuevo objeto sintáctico, formado a partir de un objeto previamente creado. De manera esquemática, esto puede definirse recursivamente de manera formal como sigue (Mota, 2013, 2014, 2015a, 2015b):

$M^{(0)}{}'(O_s) = O_s$, Def,
$M^{(n)}{}'(O_s) = M'M^{(n-1)}{}'(O_s)$ Def.

Esta definición expresa una regla que indica cómo generar objetos sintácticos. Si no aplicamos *Merge* a ningún objeto sintáctico, entonces obtenemos el mismo objeto sintáctico. Si aplicamos *n* veces *Merge*, entonces obtenemos un objeto sintáctico que es el resultado de haber aplicado *Merge* al objeto sintáctico obtenido como resultado de *n*-1 aplicaciones. Como puede apreciarse, esto es independiente de la organización interna de un objeto sintáctico generado. Así, la organización interna de expresiones como (1) *Juan canta muchas canciones* y (2) *El ratón que el gato que el perro perseguía mordió corría* es diferente, al menos en lo que respecta al nivel de análisis que atañe a los constituyentes. Así, la primera expresión no muestra auto-inclusión de constituyentes, mientras que la segunda sí muestra auto-inclusión de oraciones de relativo dentro de oraciones de relativo (tanto en el primer artículo como en el segundo el lector encontrará un análisis más detallado sobre este asunto).

Este análisis puede extenderse a un nivel de análisis más abstracto, aquel que se efectúa no en un nivel de constituyentes sino de esquemas, como es característico del modelo de Principios y Parámetros de la lingüística generativa. Moro (2008, p. 62 y ss.) indica que una estructura

recursiva es una estructura de cierto tipo X que contiene a otra estructura del mismo tipo X, y se centra en el esquema asimétrico que se aplica a toda estructura sintáctica (de ahí su carácter universal):



Siguiendo a Moro (2008, p. 75), el esquema SX –en donde X es cualquier núcleo de sintagma– se repite dentro del mismo esquema SX. En otras palabras, una estructura SX contiene el mismo esquema SX. Sin embargo, esto alude a la organización interna de una estructura. Por tanto, las propiedades predicadas de una regla y de una estructura son ciertamente independientes. Por un lado, tenemos que *Merge* ensambla, hablando en general, X, SZ = {X, SZ}, en el primer ensamble y SP, {X, SZ} = {SP, {X, SZ}} en el segundo. Por otro, tenemos la organización interna de la estructura final. Así, no hay que confundir la definición recursiva de *Merge* con la organización interna resultante de la estructura generada. Dicho de otro modo, no hay que confundir la definición recursiva del procedimiento mecánico (lo cual muestra qué hace dicho procedimiento), con la auto-inclusión propia de una estructura sintáctica resultante de su aplicación. Además, y como Chomsky ha insistido en varios trabajos, una cosa es definir recursivamente un procedimiento y otra muy diferente es implementarlo. Así, la recursión caracteriza el primer nivel, mientras que la iteración caracteriza el segundo (véase los 3 primeros artículos para sendos ejemplos de ello).

Finalmente, la esencia de la diferencia entre ambos usos, o la esencia gramatical –en términos wittgensteinianos– radica en que el uso original se emplea en reglas de substitución, y no de auto-inclusión; mientras que el segundo, aquel que tiene que ver con la organización interna de una

estructura, sea sintáctica –como en la ciencia cognitiva– o de datos –como en la ciencia de la computación–, se emplea para caracterizar la auto-inclusión de componentes dentro de otros del mismo tipo.

Por tanto, atendiendo al tema central y a los dos objetivos de la tesis señalados hasta aquí, a saber, analizar el uso de la noción de recursión en la ciencia cognitiva del lenguaje y diferenciar, al menos, entre dos usos que llevan a significados diferentes e independientes, podemos establecer que la *tesis* de este trabajo es: el uso original de recursión se aplica a la definición del procedimiento mecánico finito definido por Chomsky, dado que es una clase especial de funciones recursivas y por tanto cae dentro de la teoría de la computabilidad y que, como tal, la recursión se constituye como propiedad de una definición, esto es, una regla que permite construir otras reglas de substitución; mientras que el uso aplicado a las estructuras no responde, estrictamente hablando, al uso original, sino que se emplea como sinónimo de auto-inclusión, que caracteriza, a distintos niveles, la organización interna de las estructuras sintácticas. Dicho esto, propongo reservar la noción de recursión a su uso original, mientras que propongo aplicar 'auto-inclusión' para caracterizar la organización interna de las mencionadas estructuras.

Un asunto colateral, aunque no por ello menos importante, que considero necesario resaltar atañe al concepto de 'auto-referencia', concepto que aparece frecuentemente asociado a los de recursión y la auto-inclusión. Pese a que la auto-referencia se presenta a menudo como propiedad relacionada con la recursión y con la auto-inclusión, se trata en realidad de una característica transversal, por así decirlo, a ambas. Por consiguiente, todo aquello de lo que se predica auto-referencia no se convierte automáticamente en recursivo (en el primer artículo expongo un *addendum* sobre esta cuestión). En otras palabras, no es correcto señalar que un X es recursivo porque es auto-referente. Hay instancias de auto-referencia que no son recursivas, como por ejemplo, la conocida Paradoja de Russell, que afirma que el conjunto que contiene a todos los conjuntos que no son miembros de sí mismos, es miembro de sí mismo.

Expuestos tanto el tema central como los dos primeros objetivos de la tesis, creo conveniente presentar por qué considero que tanto Wittgenstein como Chomsky son dos autores centrales en este trabajo, sobre los que orbitan todas las reflexiones. Por lo que respecta a Chomsky, ya ha quedado patente la significativa aportación que ha hecho este autor al papel que cabe atribuir a la recursión en la facultad humana del lenguaje, entendida ésta como sistema de cómputo. Seguidamente, dedicaré unas líneas a justificar por qué Wittgenstein es también un autor que considero esencial.

En primer lugar, Wittgenstein ha trabajado el concepto de recursión en sus obras. Así, tanto en las *Observaciones filosóficas* (1975 véase por ejemplo, XIV) como en la *Gramática filosófica* (1974), podemos encontrar extensas observaciones sobre la definición por recursión. De hecho, se puede establecer una interesante relación entre tales observaciones y la obra que marca su primer período de ejercicio filosófico, a saber, el *Tractatus lógico-philosophicus* (1922). A continuación expongo un análisis en el que destaco tal relación, subrayando una serie de puntos importantes para entender la noción de algoritmo como forma general de una serie de formas. Bajo tal conceptualización también cae el procedimiento mecánico formulado por Chomsky y, por tanto, tenemos un nexo a partir del cual analizar la estrecha relación que existe entre ambos autores.

En sus *Observaciones* (XIV, §163), y en relación con el teorema de Skolem (1923), Wittgenstein indica que una expresión como 'a+(b+c) = (a+b)+c'…A(c), es concebible como una regla fundamental de un sistema. De ahí, Wittgenstein señala que: "Por supuesto que una definición no es algo que yo pueda negar. Pero entonces tampoco tiene un sentido. Es una regla en concordancia con la cual puedo proceder (o tengo que proceder)". En esa misma obra, (XIV, §164), Wittgenstein apunta que "[u]na prueba recursiva es únicamente un instructivo general para una prueba especial cualquiera. Es un letrero que indica a cada proposición [matemática] de una forma particular un camino particular para llegar a casa. Ella le dice a la proposición 2+(3+4) = (2+3)+4: 've en *esta* dirección (corre a lo largo de esta espiral) y llegarás a casa'." (cursivas en el original).

Por su parte, en la *Gramática* (1974, 32) Wittgenstein señala, por ejemplo, que una definición recursiva y una definición iterativa tienen gramáticas diferentes; esto es, se usan como reglas distintas (aunque computacionalmente sean equivalentes; es decir, aunque den el mismo valor para el mismo argumento dado).

Veamos ambos esquemas, instanciados en la función suma:

(a) Definición recursiva

(I) $u+0 = u$

(II) $u+Sx = S(u+x)$, donde S es la función sucesor.

(b) Definición iterativa

(I) $u+0 = u$

(II) $u+x = S^x u$, donde S se aplica x veces a u.

Así, Wittgenstein parece hacer una distinción similar en la *Gramática* (32), cuando señala que "es bastante claro que debe haber una "prueba" recursiva o mejor iterativa" que nos señale que así debe ocurrir con todos los números. En este sentido, Wittgenstein proporciona una regla que concuerda con la inducción matemática. Tal regla algebraica la expresa como sigue:

(I) $f(n) \times (a+b) = f(n+1)$

(II) $f(1) = (a+b)$

Por tanto, tenemos que $f(1) \times (a+b) = (a+b)^2 = f(2)$; $f(2) \times (a+b) = (a+b)^3 = f(3)$, etc. Por tanto $(a+b)^n = f(n)$. Así, "una vez que se tiene la inducción, todo ha terminado".

Dicho esto, podemos distinguir entre un proceso recursivo y otro iterativo (cf. Abelson et al., 1996). Un proceso recursivo ya ha sido mostrado en el excurso presentado unas líneas más arriba, cuando mostraba cómo computar 2+4. Una implementación iterativa, siguiendo el esquema (b) presentado unas líneas más arriba, sería como sigue:

2+4 = 2+1+1+1+1,
= 3+1+1+1,
= 4+1+1,
= 5+1,
= 6

Una implementación recursiva instancia una definición recursiva y el procedimiento invoca un valor previamente computado para un argumento menor; en cambio, mediante una implementación iterativa, una operación genera sucesivamente valores desde un argumento a otro sin usar valores previamente calculados para argumentos menores.

Sin embargo, esta no es la única observación interesante que se puede derivar de la obra de Wittgenstein. También podemos, a mi juicio, hacer una interesante aproximación a la noción de algoritmo y esto conecta directamente con su primera obra, el *Tractatus* (de esto me ocupo con mayor extensión en el cuarto artículo de esta tesis). Como muestra sirva lo siguiente. En el Tractatus (6.03), Wittgenstein presenta la forma general de un número como sigue: [1, ξ, ξ+1]. Esta forma general expresa una variable, pues el concepto de número, que para Wittgenstein es un concepto formal, se expresa mediante una variable. Tal forma general condensa una regla a partir de la cual cualquier número puede obtenerse como sucesor de otro número, siendo el número 1 –o, en otras ocasiones el 0– el inicio de la serie (esta formulación recuerda mucho los axiomas de Peano. Aunque no desarrollaré aquí esta cuestión, tal forma general puede desplegarse, por así decirlo, en los 5 conocidos axiomas de Peano). La regla a la que me refiero puede expresarse en términos de una definición recursiva como sigue:

ξ+1 = ξ',
a+(ξ+1) = (a+ξ)+1.

Dicha regla proporciona la definición recursiva de la suma. Lo que quiero resaltar aquí es que la forma general presentada más arriba fue dada

por Wittgenstein como una instancia de una serie de formas; esto es, una serie ordenada por relaciones internas (4.1252).

Por tanto, la definición recursiva ocupa un lugar importante en las definiciones de las series de formas presentadas por Wittgenstein en el *Tractatus,* como son la de los números naturales y la serie de formas que resulta de ordenar las estructuras de las proposiciones (cf. 5.2). Esta última serie de formas se instancia en la forma general de una proposición (cf. 6): $[\tilde{p}, \tilde{\bar{\xi}}, N(\tilde{\bar{\xi}})]$. Así, y como señalé más arriba, "[l]a definición recursiva es una regla para la construcción de reglas de substitución, o también el término general de una serie de definiciones [o formas]". Esto entraña la posibilidad de escribir la definición recursiva como una serie de definiciones:

1+(1+1) = (1+1)+1, 2+(1+1) = (2+1)+1
3+(1+1) = (3+1)+1,…, etc.
1+(2+1) = (1+2)+1, 2+(2+1) = (2+2)+1
3+(2+1) = (3+2)+1,…, etc.
1+(3+1) = (1+3)+1, 2+(3+1) = (2+3)+1
3+(3+1) = (3+3)+1,…, etc.

Dado que la definición recursiva también puede expresarse mediante una forma general como $[1, \xi, \xi+1]$, esto es perfectamente extrapolable para la forma general de una serie de formas, esto es, un concepto formal, una variable, expresada por Wittgenstein en el *Tractatus* (5.2522) como: [*a*, *x*, *O'x*]. Esta forma general puede definirse por recursión como sigue:

$O^{(0)}{}'(a) = a$ Def,
$O^{(n)}{}'(a) = O'O^{(n-1)}{}'(a)$ Def.

Esta definición recursiva recordará al lector la definición recursiva de *Merge*, y es que la serie de objetos sintácticos que se genera mediante aplicaciones sucesivas de *Merge* es también una serie de formas. Esto conecta no sólo a Chomsky con la noción de algoritmo y, por tanto, con los desarrollos en la teoría de la computabilidad, sino que también conecta a

Wittgenstein con la noción de algoritmo (algo que analizo en el cuarto artículo de la tesis) y, por tanto, con Chomsky. Así pues, en un plano formal, Wittgenstein y Chomsky consideran como central en sus obras a las series de formas; en definitiva, a los algoritmos.

Todas las consideraciones que he hecho hasta aquí tienen que ver, como el lector podrá haber adivinado ya, con la investigación formal sobre el concepto de recursión. A continuación, haré una breve exposición sobre lo que podemos llamar el problema empírico de la recursión. El análisis de este problema y las consecuencias que se derivan del mismo para la investigación empírica (como opuesta a formal) de la recursión constituye el tercer objetivo de la presente tesis. En lo que sigue, analizaré qué relación podemos establecer entre el anterior análisis conceptual sobre la noción de recursión y el papel que desempeña la investigación empírica en todo esto; Con ello, me propongo analizar, por un lado, si la investigación empírica puede ayudar a clarificar el concepto de recursión y, por otro, si la noción de recursión tiene aplicación en la investigación empírica. Nótese que estos dos aspectos del problema son independientes, ya que el hecho de que la investigación empírica sirva o no para clarificar el concepto de recursión análisis nada tiene que ver con el modo en que se aplica este concepto en la investigación empírica como herramienta.

En lo que respecta a la primera cuestión, creo que la investigación empírica no es el tipo de investigación requerida ni oportuna para la clarificación y el análisis del concepto de recursión. Como ya he señalado más arriba, la propiedad de la recursión se constituye, por así decir, en una regla. Dicho de otro modo, la recursión se muestra en el uso que se hace una regla, de una determinada forma de proceder. Como tal, la regla no afirma o asevera nada, y por tanto las expresiones que calificamos como reglas no han de ser confrontadas con ninguna realidad.

En el análisis de este punto que el lector encontrará a continuación (y que se desarrolla principalmente en los artículos segundo y tercero, aunque esta cuestión es transversal a los cuatro primeros), conviene distinguir entre expresiones como:

(a) 2+2 = 4

(b) El sujeto S sigue la regla ‘2+2=4’

En el caso de (a) estamos ante una regla, y esa regla no *dice* nada sobre el mundo, mientras que (b) *dice* que un sujeto S *usa* la regla (a). Así, (b) se *usa* como una proposición empírica, mientras que (a) se *usa* como una regla. En este sentido, (b) presupone (a). Lo que quiero señalar aquí es que no descubrimos objetos matemáticos, como puedan ser reglas, algoritmos, números o series, los creamos (cf. Wittgenstein 1978, I, §168; Monk, 1990). Por ello, creo que la investigación empírica no debe ir dirigida a fundamentar o justificar la noción de recursión *en sí*, podríamos decir, sino más bien a determinar si *en* la investigación empírica se pueden usar herramientas que hagan uso de la definición por recursión. En este segundo punto se sitúa el trabajo empírico que se incluye en esta tesis (y que será introducido unas líneas más abajo). A este respecto, y obtenidos unos datos, un algoritmo definido recursivamente se *aplica* para dar sentido a tales datos. Ahora bien, conviene subrayar, por un lado, que el algoritmo *no* se ha descubierto, se ha creado o inventado y, por otro, que el algoritmo puede o no tener aplicación (esta es *la* cuestión empírica que tenemos entre manos). Por consiguiente, el algoritmo no es verdadero o falso; es decir, no describe ningún estado de cosas. Dicho de otro modo, y en relación con el dominio que se examina en este trabajo, no *describe* cómo se relacionan determinados objetos sintácticos en la cabeza del sujeto ni *describe* el nivel neurológico con un determinado grado de abstracción. Si ese algoritmo mostrara ser inaplicable porque los datos, en otros experimentos o en otras investigaciones, fueran en ‘otra dirección’, no sería por ello falso, igual que la demostración de que sí es aplicable no lo convierte en verdadero. El punto crucial está, a mi entender, en concebir que el algoritmo no describe nada, sino más bien que es aplicable o inaplicable. En este caso, el algoritmo no es un conjunto de proposiciones que describen de manera verdadera cómo ciertos objetos (i.e., los objetos sintácticos) se relacionan entre sí por medio de una operación (i.e., *Merge*), sino de reglas, reglas de uso que indican cómo efectuar cálculos con objetos construidos (i.e., los

objetos sintácticos) dentro de un cálculo consistente en la aplicación de una operación (i.e., *Merge*).

En consecuencia, es legítimo en este contexto hablar de lo mental, pero desde mi punto de vista, cuando se habla de lo mental, al menos bajo la perspectiva computacional, no se está hablando ni de una región más o menos acotada del mundo, sea éste empírico o platónico (Chomsky rechaza esta última opción, mientras que sostiene la primera; cf. Chomsky, 1980), ni de una descripción abstracta del cerebro. En mi opinión, el vocabulario de *lo mental* muestra una gramática, esto es, unas reglas para el uso de expresiones tales como 'representación mental', 'objeto sintáctico',… etc.[10] Por otro lado, y en esta misma línea, un mecanismo computacional como el que Chomsky define, caracteriza de manera abstracta una capacidad cognitiva como el lenguaje, y esto no ha de presentar problema alguno si lo que se quiere mostrar con ello es el uso que se hace de 'lenguaje' dentro de su concepción. Ahora bien, el problema, a mi juicio, radica en que esa caracterización formal, abstracta se tome como descriptiva de un estado de cosas. En este nivel formal, no se describe nada relacionado con el cerebro, a lo sumo se describe el uso de conceptos formales como el de 'objeto sintáctico'. Por el contrario, las descripciones en el nivel neurológico pueden tener que ver con cómo determinadas regiones del cerebro se activan según qué actividades realicen, pero eso no es describir el mecanismo computacional en un nivel de análisis más bajo; es describir el cerebro. En este sentido, una descripción del nivel neurológico presupone una gramática diferente, en la que, por ejemplo, 'objeto sintáctico' no tiene el mismo uso (si es que tiene alguno) que en el nivel formal. Por tanto, mis reservas con lo mental tienen que ver (como verá el lector en los artículos segundo y tercero) con lo rápido que se ha asumido que el mecanismo computacional se realiza neurológicamente, dando, por decirlo así, un salto ontológico que

[10] Estas observaciones sobre lo mental no significa que lo mental sea reducible a la conducta (más bien, ésta es considerada un criterio de lo mental); es decir, la gramática de lo mental no es la gramática del ejercicio particular de la conducta. Por otro lado, lo mental no se puede reducir a lo neuronal o cerebral, por ello, tampoco puede asimilarse a una propuesta materialista; es decir, la gramática de lo mental no es la gramática de la activación cerebral. Por último, tampoco sostengo una postura dualista cartesiana. Lo que propongo, entonces, es que lo mental está constituido por un mapa conceptual integrado por conceptos psicológicos. Así, la investigación de lo mental no es psicológica o experimental, sino conceptual o gramatical.

no está, en mi opinión, justificado por el propio Chomsky. En otras palabras, Chomsky no justifica cómo es posible que una clase especial de funciones recursivas se realice neurológicamente. Esa asunción no reduce el salto que hay entre un análisis computacional del lenguaje y un análisis neurológico. Así, que la aritmética, como capacidad o habilidad humana se sustente neurológicamente es una cuestión muy diferente de asumir que, por ello, la teoría de números sea una descripción abstracta del nivel neurológico.

1.2. Presentación de publicaciones que integran la tesis

La presente tesis doctoral consta de cinco artículos, los cuatro primeros publicados o aceptados para su publicación y el quinto preparado para su envío. A continuación, expondré, con el propósito de que el lector se haga una idea, siquiera general, de cada uno de los artículos compilados en esta tesis, un resumen del objetivo principal y los contenidos de cada uno de ellos.[11]

La relación de artículos se inicia con el titulado "*Sobre el concepto de recursión y sus usos*". En este artículo se analizan dos usos del término 'recursión', a saber: el de la definición por recursión y el de auto-inclusión. Para ello, se transita por diferentes dominios de conocimiento, como son la lógica matemática, la ciencia de la computación y la ciencia cognitiva. Asimismo, se establece una diferenciación semántica entre las nociones de 'recursión' y 'auto-referencia'. En este sentido, se propone reservar el concepto de recursión para su uso original, esto es, como método de definición y, por tanto, como propiedad de una regla, mientras que la organización interna de estructuras que contienen un X dentro de otro X del mismo tipo puede caracterizarse, sin menoscabo conceptual alguno,

[11] Quizá pueda llamar la atención que todas las revistas pertenecen a ámbitos distintos al de la psicología. Por ello, quisiera resaltar, primero, su relevancia en los campos a los que pertenecen. Las revistas de Filosofía General y de Filosofía de la Ciencia en donde aparecen estos artículos han sido recomendaciones de expertos en la materia. Lo mismo puedo decir en relación con la revista de Lingüística (teórica y aplicada). Por otra parte, el trabajo realizado en esta sección no tiene que ver con consideraciones empíricas o experimentales, sino más bien, con problemas conceptuales, que requieren un análisis gramatical. Por ello, creo idónea la pertinencia de las revistas seleccionadas a la luz del tipo de trabajo realizado.

mediante la noción de auto-inclusión. Aunque ambas nociones sean tipos de auto-referencia, no hay que identificar la recursión con la auto-referencia, dado que no son nociones coextensivas.

El segundo de los artículos, titulado *"¿Por qué se usa 'recursión' cuando se quiere significar 'auto-inclusión'?: clarificaciones conceptuales sobre la recursión en el programa chomskiano*", trata de aplicar el análisis expuesto en el artículo anterior al dominio de la ciencia cognitiva del lenguaje y, más concretamente, al programa chomskiano. Así, el objetivo principal es disociar (doblemente) la noción de recursión de la de auto-inclusión. Además, en la parte final de este artículo se responde a una serie de cuestiones que anticipan parte del análisis del tercer artículo, cuestiones que tienen que ver, por ejemplo, con la eventual objeción que, a juicio de algunos, supone la existencia de lenguas sin estructuras recursivas (o, mejor dicho, sin estructuras con *auto-inclusión*) o con la realidad psicológica de la recursión. En el primer caso, no hablamos de la recursión *en* la facultad del lenguaje, *en* el lenguaje entendido como un mecanismo computacional. Hablamos, más bien, de la organización interna de las estructuras que genera o con las que opera. Así, el mecanismo computacional puede definirse recursivamente sin que sea una objeción que las estructuras que genera no presenten auto-inclusión. Este ejemplo, precisamente, muestra que 'recursión' y 'auto-inclusión' tienen *usos diferentes*.

El tercer artículo, titulado "*Sobre el anti-realismo de Wittgenstein y su aplicación al programa chomskiano*", recoge, por así decir, parte de los análisis presentados en los dos primeros artículos. Del primer artículo recupera parte de la investigación formal y del segundo retoma los análisis derivados de la aplicación de ese análisis formal al programa chomskiano. En este artículo se enfatiza que los procedimientos generativos (i.e., los algoritmos) se construyen, no se descubren y, por tanto, no son entidades u objetos independientes con una existencia propia (i.e., con una ontología). Es en este sentido en el que hablo de anti-realismo. Por ello, el mecanismo computacional chomskiano es una construcción. Esto quiere decir, por un lado, que no se ha descubierto una nueva región del mundo y, por otra, que el mecanismo computacional no expresa verdades que puedan describirse

con un vocabulario neurológico pero en un nivel de abstracción más bajo. Tal mecanismo caracteriza de manera abstracta al lenguaje y esto ha de entenderse como que tal caracterización muestra lo que se entiende por 'lenguaje' en el programa chomskiano. Así, lo mental, en este nivel computacional, ha de entenderse como una caracterización abstracta de la facultad del lenguaje que, por su parte, resulta en un mecanismo computacional que no es sino una clase especial de funciones recursivas y que, como tal, es susceptible de una investigación matemática. La gramática de esta indagación matemática es diferente, pero no más abstracta, que la gramática neurológica, aplicada esta última a una investigación neurocientífica, no matemática.

El cuarto artículo, titulado "*¿Qué es un algoritmo? Una respuesta desde la obra de Wittgenstein*", cierra la investigación formal considerando que la noción de algoritmo, transversal a todos los artículos previos, es un concepto formal y, como tal, se expresa mediante una variable; esto es, mediante la forma general de una serie de formas. Por ello, este artículo concluye que la supuesta dualidad entre *máquinas de estados abstractas* (como la máquina de Turing) y *recursores* (una descripción recursiva construida a partir de operaciones tomadas como primitivas; ver *supra*) no responde a la pregunta de qué es un algoritmo. Ambas respuestas son instancias de una serie de formas. Dicho de otro modo, un algoritmo no es *o* una máquina de estados abstractas *o* un recursor (esto es, las diferencias intensionales no son esenciales para determinar qué es un algoritmo, pues ambas propuestas caen bajo esa noción formal); *ambas* respuestas instancian una serie de formas. Por ello, la dualidad se diluye, por así decir, en el análisis gramatical expuesto en dicho trabajo.

***

En relación con el trabajo experimental, reflejado en el artículo que lleva por título "*Parsing complex phrases: Recursion, hierarchical structure, and memory load*" creo conveniente hacer las siguientes

observaciones, en aras de una mejor organización de la información para facilitar al lector, en la medida de los posible, su comprensión.

Resumen

En este artículo presentamos dos experimentos en español (ambos divididos en dos subexperimentos) diseñados para descubrir qué clase de procesos subyacen al análisis sintáctico en tiempo real de sintagmas nominales complejos con constituyentes anidados. Para ello, se empleó un paradigma de "detección de clicks" asociada a una tarea de comprensión oral de enunciados. El estímulo distractor era un tono ubicado en dos posiciones posibles de cada enunciado: al término del primer o tercer nombre del SN complejo. Esta tarea se aplicó, en el primer experimento, a dos tipos de estructuras, sintagmas preposicionales subordinados insertos en un sintagma nominal y sintagmas nominales coordinados, en oraciones compuestas por palabras (subexperimento 1A) o pseudo-palabras (subexperimento 1B). Esto se contrastó, en el segundo experimento, con el procesamiento de listas no estructuradas de palabras (o pseudo-palabras) con la misma tarea (subexperimentos 2A y 2B, respectivamente). Los resultados sugieren que, a diferencia de lo que ocurre con las secuencias desprovistas de estructura, ambas clases de estructuras (tanto si contienen palabras como pseudopalabras) comparten demandas de procesamiento parecidas y, por consiguiente, el procesamiento de ambas se puede caracterizar mediante un algoritmo definido recursivamente en un nivel abstracto de análisis.

Objetivo general: analizar las operaciones subyacentes a los procesos de análisis sintáctico (*parsing*) de enunciados complejos de diversos tipos en hispano-hablantes, con el fin de comprobar hasta qué punto dichas operaciones se podían caracterizar mediante un algoritmo definido por recursión.

Objetivos específicos:

(1) Contrastar el procesamiento de dos tipos de estructuras, sintagmas preposicionales subordinados insertos en un sintagma nominal complejo y sintagmas nominales coordinados, en oraciones compuestas por palabras o pseudo-palabras (Experimento 1).[12] Un ejemplo de cada es como sigue:

Oración subordinada compuesta por palabras:

Me pareció que los dibujos del comic de la revista del domingo eran muy graciosos.

Oración coordinada compuesta por palabras:

Me pareció que el título, la trama los diálogos y los chistes eran muy graciosos.

Oraciones subordinadas compuestas por pseudo-palabras:

Me pareció que las nobesas del brucio de la filoca del petiso eran muy graciosas.

Oraciones coordinadas compuestas por palabras:

Me pareció que las nobesas, el brucio, la filoca y el petiso eran muy graciosos.

(2) Contrastar los datos del experimento 1 con el procesamiento de listas no estructuradas de palabras (o pseudo-palabras). Un ejemplo de cada es como sigue:

Lista de palabras:

Tienes que recordar: título, trama, diálogos, chistes.

Lista de pseudo-palabras:

Las pseudopalabras son: nobesas, brucio, filoca, petiso.

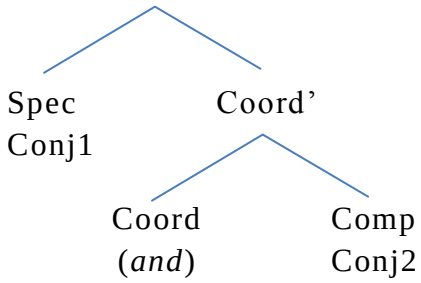
Consideraciones previas a las hipótesis de trabajo:

La descripción estructural de los sintagmas nominales complejos con sintagmas preposicionales se ajusta al esquema especificador-núcleo-complemento, siendo los sintagmas preposicionales anidados complementos del núcleo nominal que encabeza el SN complejo. En cambio, en la

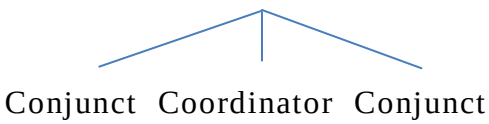
[12] Este experimento se divide, a su vez, en dos subexperimentos. El subexperimento 1 contrasta las estructuras mencionadas en oraciones compuestas por palabras. El subexperimento 2 lo hace en oraciones compuestas por pseudo-palabras (i.e., en pseudo-oraciones).

literatura especializada hay actualmente un debate en torno al tipo de estructura que subyace a la coordinación gramatical, ya sea en sintagmas o en oraciones. Este debate es importante de cara al análisis sintáctico en el procesamiento. En primer lugar, ambos tipos de sintagmas nominales complejos comparten (o pueden compartir) una estructura jerárquica binaria (1a en el ejemplo). En concreto, que las estructuras coordinadas presentan una estructura jerárquica binaria ha sido sostenido por varios autores (véase Johannessen, 1998; Camacho, 2003; Zhang, 2010 para más detalles). Bajo esta caracterización jerárquica y asimétrica, y en términos de la Teoría de la X con Barra, un sintagma nominal coordinado se representa como una proyección funcional (llamado Sintagma Coordinación; ver *infra*) que tiene como núcleo la conjunción *y* (en el ejemplo, *and*), como *especificador* el primer miembro de la conjunción, y, por último, el segundo miembro como su *complemento*. En la actualidad, esta estructura es referida a la clásica estructura 'plana', representada como una estructura con una ramificación ternaria (1b).

(1) a.



b.



Hay, sin embargo, un análisis alternativo al anterior en relación con las estructuras jerárquicas binarias de sintagmas nominales coordinados, que contempla el sintagma-*y* como un adjunto del primer miembro de la coordinación. Bajo esta perspectiva, los nombres en un sintagma nominal coordinado aparecen como adjuntos unos con otros, pero no forman la configuración anteriormente expuesta de especificador-núcleo-complemento; estando, así, en una relación de concatenación.

Dicho esto, nuestro trabajo abordó la diferencia entre estructuras subordinadas y coordinadas desde dos enfoques, uno en el que el significado

podía jugar un papel importante (subexperimento 1A) y otro en el que las estructuras eran procesadas sin una influencia del significado léxico tan prominente como en el primer caso (subexperimento 1B). Se decidió esta estrategia debido a que, en principio, parece razonable suponer que hay diferencias en las demandas de procesamiento a la hora de procesar sintagmas incrustados frente a sintagmas coordinados. Así, los sintagmas nominales complejos con sintagmas preposicionales incrustados parecen, en principio, requerir, un procesamiento jerárquico anidado, manteniéndose activos una serie de rasgos morfológicos (denominados *φ-features* –rasgos-*φ*), como son los rasgos de género y número, en el caso de los nombres en español, que determinan las operaciones de concordancia del nombre con otros constituyentes dentro y fuera del sintagma nominal (i.e., con adjetivos, en el primer caso, y con el verbo, en el segundo), necesarios para establecer las relaciones sintácticas de dependencia a larga distancia entre el núcleo del sintagma nominal complejo y el predicado, integrado en un sintagma verbal. Esto puede implicar operaciones diferidas complejas con auto-llamadas recurrentes cada vez que un objeto sintáctico es recibido y analizado.

Es importante tener presente que el procesamiento recursivo no se reduce al procesamiento de estructuras jerárquicas como las recién mencionadas. Es decir, las características de una generación recursiva (como en el caso de *Merge*, ver *supra*) o de un proceso recursivo no devienen del tipo de estructura procesada, sino de cómo se llevan a cabo las operaciones (ver *supra* la caracterización de un proceso recursivo). Sobre este punto volveré más abajo.

En contraste con lo expuesto en relación con las estructuras subordinadas, parece que, en principio, el procesamiento de sintagmas nominales coordinados requiere añadir, en cada paso del proceso, un nuevo sintagma nominal al sintagma nominal bajo construcción. Este proceso también puede implicar operaciones diferidas, aunque basadas en un tipo de información diferente que en el caso precedente, dado que los rasgos del sintagma nominal complejo están dados por la pluralidad del conjunto denotado por el sintagma nominal (por ejemplo, ‘los bichos, las

plantas,….'), no teniendo que realizar un cotejo de rasgos-φ del núcleo nominal del SN para establecer una concordancia a larga distancia en este caso. Así, podría argumentarse que el procesamiento de sintagmas incrustados en un sintagma nominal complejo impone demandas más duras a la memoria de trabajo que el procesamiento de sintagmas nominales coordinados.

Por tanto, si las demandas de memoria sobre los procesos (diferidos) de concordancia prevalecen, entonces deberíamos esperar diferencias entre el procesamiento de sintagmas subordinados y coordinados. Por otro lado, si el análisis sintáctico es más sensible a la configuración sintáctica, o si la configuración sintáctica prevalece sobre otros rasgos sintácticos implicados en el procesamiento necesarios para la concordancia, entonces no deberíamos encontrar grandes diferencias en el procesamiento de ambas estructuras.

Como no hay modo alguno de saber a priori qué tipo de información prevalece sobre la otra, nuestro diseño experimental trata de cubrir ambas posibilidades, empleando oraciones con palabras y oraciones con pseudo-palabras con, evidentemente, ambos tipos de estructuras. En este sentido, puede decirse que nuestro estudio es exploratorio.

Hipótesis de trabajo

Siguiendo lo dicho anteriormente, se pueden enunciar cuatro hipótesis, las dos primeras alternativas, relativas, respectivamente, a la comparación entre el procesamiento de sintagmas con subordinación y el de sintagmas coordinados (hipótesis 1 y 2), al contraste entre estructuras sintácticas con y sin contenido léxico (hipótesis 3) y a la diferencia entre enunciados lingüísticos dotados de estructura (sintagmas) y desprovistos de ella (series de palabras) (hipótesis 4).

(1) ***Hipótesis 1*:** Habrá diferencias entre el procesamiento de estructuras subordinadas y coordinadas en el experimento 1 (realizado con oraciones), con independencia de su composición léxica, lo que se reflejará en un patrón diferente de los tiempos de reacción al tono distractor en ambos subexperimentos.

Si hay diferencias entre ambas estructuras en el experimento 1, realizado con oraciones, entonces cabe pensar que los requisitos sintácticos para la concordancia gramatical tienen un peso importante en el procesamiento. Así pues, si el patrón en los tiempos de reacción es diferente, se podrá concluir que la tarea empleada en los experimentos es sensible al proceso de mantenimiento o recuperación de los rasgos-φ del núcleo nominal, que supuestamente es más complejo en sintagmas nominales compuestos de sintagmas subordinados que en sintagmas nominales coordinados.

(2) ***Hipótesis 2*:** No habrá diferencias entre el procesamiento de estructuras subordinadas y coordinadas en ninguno de los subexperimentos (1A y 1B), mostrando las dos estructuras en ambos casos un patrón similar de tiempos de reacción.

Si no hay diferencias entre ambas estructuras en los subexperimentos del experimento 1, entonces cabrá pensar que la tarea empleada en el experimento es sensible únicamente a la configuración sintáctica de las dos estructuras examinadas, y no a otras operaciones gramaticales diferidas, implicadas en el establecimiento de la concordancia del SN complejo con el verbo de la oración.

(3) ***Hipótesis 3*:** No se espera que haya diferencias entre el procesamiento de cada tipo de estructura en función de su composición léxica, es decir, el patrón de resultados será el mismo con independencia de si los sintagmas objeto de estudio están compuestos de piezas léxicas o de pseudopalabras sin sentido.

La confirmación de esta hipótesis serviría para demostrar el carácter puramente estructural de los procesos subyacentes a la tarea de comprensión, sin posible "contaminación" de estos procesos por factores léxico-semánticos.

(4) ***Hipótesis 4*:** Habrá diferencias entre el procesamiento de ambas clases de estructuras (tanto si contienen palabras como pseudopalabras) y el procesamiento de listas no estructuradas de palabras (o pseudo-palabras). Esta comparación se llevó a cabo en el experimento 2.

Si hay diferencias entre el procesamiento de ambas clases de estructuras (tanto si contienen palabras como pseudopalabras) y el procesamiento de listas no estructuradas de palabras (o pseudo-palabras), entonces cabe pensar que los resultados obtenidos no se deben meramente al incremento progresivo de la información y una consiguiente mayor carga de memoria a medida que avanza el procesamiento, sino que los efectos registrados obedecen a factores asociados al procesamiento sintáctico.

Resultados:

En relación con las hipótesis (1) y (2), NO se han obtenido diferencias significativas en el procesamiento de estructuras subordinadas y coordinadas en el subexperimento 1A (esto es, con oraciones), lo que se ha reflejado en un patrón similar de tiempos de reacción. Este patrón consiste en tiempos de reacción más lentos al estímulo distractor cuando éste se sitúa en una posición más tardía (y jerárquicamente más profunda) en la estructura del enunciado.

En relación con la hipótesis (3), NO hubo diferencias en el procesamiento de estructuras subordinadas y coordinadas en función de su composición léxica, es decir, ambas estructuras reflejando un patrón similar de tiempos de reacción en los subexperimentos 1A y 1B del primer experimento.

Estos resultados, tomados conjuntamente, parecen apuntar a un mayor peso de la información sintáctica que semántica en la tarea experimental, tal y como se apuntaba más arriba.

En lo tocante a la hipótesis (4), SÍ se registraron diferencias en el patrón de tiempos de reacción entre las estructuras subordinadas y coordinadas (compuestas por palabras o por pseudopalabras), por un lado, y el procesamiento de listas no estructuradas de palabras (o pseudo-palabras), por otro.

Estos resultados sugieren que, a diferencia de lo que ocurre con las secuencias desprovistas de estructura, las estructuras subordinadas y coordinadas comparten demandas de procesamiento parecidas, lo que, a su vez, parece indicar que el procesamiento que subyace a la tarea está

determinado más por factores estructurales, siendo los aspectos semánticos secundarios a la información sintáctica. Esto no pretende sostener que los rasgos semánticos carezcan de importancia, sino simplemente que el procesador parece ser más sensible a los aspectos sintácticos en lo tocante a la tarea empleada en los experimentos.

La elección del algoritmo:

El algoritmo construido, no descubierto, para caracterizar el procesamiento crea, por así decir, la forma de los hechos. ¿Qué quiere decir esto? Que la regla definida mediante el presente algoritmo es autónoma, pero encuentra su aplicación en los datos obtenidos.

Así, el patrón de tiempos de reacción obtenido tanto en el subexperimento 1A como en el subexperimento 1B del primer experimento puede resumirse indicando que el tiempo de reacción obtenido en la detección del 'click' (tono) situado en una posición temprana del sintagma nominal complejo (ver artículo 5 para más detalles) es más rápido (en ambos subexperimentos y en ambas estructuras) que el tiempo de reacción obtenido en la detección del 'click' situado en una posición más tardía.

Se proponen dos posibles algoritmos:

(a) Algoritmo definido recursivamente

$$\sigma^{(0)}(a) = a$$

$$\sigma^{(n)}(a) = \sigma\text{'}\sigma^{(n-1)\text{'}}(a)$$

En donde 'σ' se refiere a la variable operacional, cuyos valores son aquellas operaciones gramaticales realizadas, '*a*' es una variable cuyos valores son objetos sintácticos y '$\sigma^{(n-1)\text{'}}(a)$' está por un objeto sintáctico procesado previamente que $\sigma^{(n)}(a)$.

Así, cada nuevo objeto sintáctico se procesa haciendo uso de objetos sintácticos previamente procesados (i.e., computados):

Ejemplo

Consideremos qué ocurre cuando el objeto sintáctico '*remolque*' es ensamblado con un objeto sintáctico previamente computado, a saber '*la rueda del*' para obtener el sintagma '*la rueda del remolque*' (este ejemplo está tomado del artículo 5). Así, el proceso recursivo es como sigue:

(1) $\mathfrak{G}^{(4)}(la) = \mathfrak{G}_{remolque}{}'\mathfrak{G}^{(3)}{}'(la)$,

La rueda de el remolque = {la rueda de el}, remolque

(2) $\mathfrak{G}_{remolque}{}'\mathfrak{G}^{(3)}{}'(la) = \mathfrak{G}_{remolque}{}'\mathfrak{G}_{el}{}'\mathfrak{G}^{(2)}{}'(la)$,

{la rueda de el}, remolque = {{la rueda de},el}, remolque

(3) $\mathfrak{G}_{remolque}{}'\mathfrak{G}_{el}{}'\mathfrak{G}^{(2)}{}'(la) = \mathfrak{G}_{remolque}{}'\mathfrak{G}_{el}{}'\mathfrak{G}_{de}{}'\ \mathfrak{G}^{(1)}{}'(la)$,

{{la rueda de},el}, remolque = {{{la rueda}}, de}, el}, remolque

(4) $\mathfrak{G}_{remolque}{}'\mathfrak{G}_{el}{}'\mathfrak{G}_{de}{}'\mathfrak{G}^{(1)}{}'(la) = \mathfrak{G}_{remolque}{}'\mathfrak{G}_{el}{}'\mathfrak{G}_{de}{}'\mathfrak{G}_{rueda}{}'\mathfrak{G}^{(0)}{}'(la)$.

{{{la rueda}}, de}, el}, remolque = {{{{la}, rueda}, de}, el}, remolque

(b) Algoritmo iterativo

$\mathfrak{G}^{(1)}(a) \rightarrow \mathfrak{G}^{(1)}(a)$,

$\mathfrak{G}^{(1)}(a), b \rightarrow \mathfrak{G}^{(2)}(a)$,

…

$\mathfrak{G}^{(n)}(a),\ b \rightarrow \mathfrak{G}^{\,(n+1)}{}'(a)$

En donde '$\mathfrak{G}$' se refiere a la variable operacional cuyos valores son aquellas operaciones gramaticales realizadas, 'a' es una variable cuyos valores son

objetos sintácticos, b es otra variable cuyos valores son también objetos sintácticos y $\mathfrak{G}^{(n+1)\prime}(a)$ es el objeto sintáctico obtenido al añadir a '$\mathfrak{G}^{(n)}(a)$' el objeto sintáctico '*b*'. La flecha indica la relación de 'sustituible por' (no indica una relación diferente del signo '='). Evidentemente hay una equivalencia computacional entre un implementación recursiva e iterativa (lo cual puede comprobarse en la Introducción a los artículos y en los artículos mismos; por ejemplo en el artículo 1).

En este caso, un nuevo valor *no* se computa haciendo uso de valores previamente computados. En cambio, del ensamble de un nuevo objeto ('*b*') al objeto que en ese momento se ha construido ('$\mathfrak{G}^{(n)}(a)$') se obtiene un nuevo objeto sintáctico ('$\mathfrak{G}^{(n+1)\prime}(a)$').

Ejemplo

Veamos cómo se construye el objeto sintáctico '*la rueda de el remolque*'. Así, el proceso iterativo es como sigue:

(1) $\mathfrak{G}^{(1)}(la) \rightarrow \mathfrak{G}^{(1)}(la)$,

La

(2) $\mathfrak{G}^{(1)\prime}(la)$, *tire* $\rightarrow \mathfrak{G}^{(2)\prime}(la)$,

La rueda

(3) $\mathfrak{G}^{(2)\prime}(la)$, *de* $\rightarrow$ $\mathfrak{G}^{(3)\prime}(la)$,

La rueda de

(4) $\mathfrak{G}^{(3)\prime}(la)$, *el* $\rightarrow \mathfrak{G}^{(4)\prime}(la)$,

La rueda de el

(5) $\mathfrak{G}^{(4)'}(la)$, *remolque* $\rightarrow$ $\mathfrak{G}^{(5)'}(la)$,

La rueda de el remolque

La elección del algoritmo (a) frente al (b) está basada en los patrones de tiempos de reacción obtenidos. Así, si el procesamiento hubiera sido del tipo expuesto en (b), y aun reconociendo un incremento en la carga de memoria, el proceso es secuencial, se pasa de un objeto sintáctico a otro añadiendo en cada caso un elemento más. Sin embargo, lo que hay que hacer en cada paso es lo mismo, a saber, cotejar un objeto sintáctico con el que es recibido; es decir, cotejar $\mathfrak{G}^{(n)}(a)$ con b. Así, aunque en el paso 4 haya más carga de memoria que en el 2, porque el objeto sintáctico contiene más información, en el paso 4 no se requiere realizar más operaciones que en el paso 2, y mucho menos se requieren más operaciones diferidas. El algoritmo iterativo hubiera sido la primera elección si no se hubieran registrado diferencias significativas en los tiempos de respuesta al tono distractor entre las posiciones temprana y tardía. Sin embargo, como sí las ha habido, cabe pensar que el procesamiento no es secuencial, como en el caso del ejemplo (b), sino que parece incrementase con cada nuevo ensamble, lo que sugiere que en cada paso del proceso se da una reactivación de objetos sintácticos previamente computados. Las limitaciones, sobre todo de la tarea, a la hora de establecer definitivamente estos resultados se discuten en el siguiente apartado, el de discusión general.

Discusión general:

Los experimentos expuestos en el artículo 5 muestran que los oyentes necesitan más tiempo para detectar el click incrustado dentro de un sintagma nominal complejo en una tarea de comprensión de oraciones cuando el tono está localizado más profundamente en la estructura (i.e., en posiciones posteriores dentro del sintagma nominal) que cuando el tono se localiza en posiciones anteriores (i.e., más tempranas).

Los efectos, como he argumentado más arriba, parecen ser de naturaleza sintáctica, dada su ocurrencia en oraciones constituidas por

pseudo-palabras (i.e., pseudo-oraciones), habiendo así minimizado, en la medida de lo posible, el efecto del significado léxico. Además, y apoyando estos hallazgos, el patrón opuesto de tiempos de reacción fue hallado cuando el tono estaba situado en cadenas desestructurada de palabras (y pseudo-palabras). Esto además, parece mostrar que la sensibilidad de los participantes a los estímulos distractores viene determinada por operaciones concernientes al análisis sintáctico y no por factores puramente perceptivos, como por ejemplo, la posición serial del estímulo perceptivo en la serie.

Como también he señalado, una posible explicación es que el ensamble de un objeto sintáctico, por ejemplo, el sintagma preposicional '*del camión*', en '*la rueda*$_{N1}$ *del remolque* $_{N2}$ *del camión*$_{N3}$', cuyo núcleo '*la rueda*', implica un incremento en términos de costes del procesamiento asociado a la reactivación de objetos sintácticos previamente computados; a saber, los sintagmas nominales y preposicionales analizados anteriormente tales como '*la rueda del remolque*'. A esta descripción subyace una regla recursiva, la cual es aplicada para dar sentido a los datos obtenidos. Así que, de nuevo, no se ha descubierto nada con respecto a la naturaleza del procesamiento; más bien, se está mostrando la utilidad de una herramienta formal (en este caso, la definición por recursión) para dar cuenta de unos datos.

Esta herramienta también sirve para dar sentido a los datos obtenidos en relación con la condición experimental relativa a los sintagmas nominales coordinados. Así, el ensamble de una conjunción en sintagmas nominales complejos, tal como en el caso de *'(y) el freno'* en '*el cambio*$_{N1}$ *(y) el embrague*$_{N2}$ *(y) el freno*$_{N3}$', más allá de si es un complemento o un adjunto (véase *supra* la discusión en torno a esta cuestión, así como su análisis en el artículo 5), supone la reactivación de sintagmas nominales conjuntos previamente procesados.

Así, tomados conjuntamente, los datos relativos a los sintagmas subordinados y los relativos a los sintagmas coordinados sugieren que hay una configuración jerárquica análoga en ambas estructuras. Si se asume esta similitud estructural, parece ser que el procesador está sujeto a las mismas

restricciones (o, en todo caso, a restricciones muy similares) en cuanto a carga de memoria se refiere cuando procesa ambos tipos de estructuras.

De este modo, y resumiendo lo dicho hasta aquí, el procesador comienza codificando el primer nombre de un sintagma nominal complejo, etiquetándolo como el sujeto de una cláusula subordinada, lo cual genera la expectativa de un predicado. Esta expectativa permanece activa a través de la serie de constituyentes (esto es, de sintagmas preposicionales y nominales) que siguen al primer sintagma nominal, hasta que se satisface con la llegada del predicado. Esto tiene efectos en cuanto a carga de memoria se refiere, haciéndose patentes tales efectos en los tiempos de reacción, que se incrementan en los tonos situados en posición tardía con respecto a los situados en posición temprana. Esta descripción del proceso es congruente con modelos actuales de procesamiento, los cuales atribuyen a la memoria de trabajo un importante papel en el procesamiento de oraciones (véase Lewis et al., 2006), tanto en relación con la carga de memoria como en relación con efectos de interferencia.

Por último, podemos establecer las siguientes conclusiones: (1) el patrón de tiempos de reacción a tonos o clicks incrustados en estructuras sintácticas complejas como las empleadas en este trabajo experimental indica que la carga de memoria se incrementa en el curso del procesamiento, tanto si se trata de constituyentes subordinados como coordinados; (2) tal incremento puede ser considerado como una indicación de que el procesador mantiene activos los constituyentes codificados, de tal modo que realiza operaciones sintácticas diferidas (encaminadas, por ejemplo, a establecer concordancias entre sujeto y verbo) en fases posteriores del procesamiento; (3) si este fuera el caso, la construcción de un algoritmo definido recursivamente podría ser una herramienta útil para dar cuenta de los datos obtenidos. Por consiguiente, y dado que la actuación lingüística puede entenderse como una conducta gobernada por reglas, podría decirse que los participantes están siguiendo una regla recursiva. No obstante, una explicación y/o descripción más fina de los procesos subyacentes, por ejemplo, en relación con la carga y los efectos de interferencia del material procesado, o en relación con la aclaración de si el procesamiento requiere o

no operaciones diferidas, supone tanto diseñar como aplicar tareas y materiales diferentes. De este modo, quizá se llegué a una representación sinóptica más satisfactoria del papel de la recursión en el procesamiento sintáctico.

1.3. Referencias

Abelson, H. & Sussman, G. J. with Sussman, J. (1996). *Structure and interpretation of computer programs*. Cambridge, MA: MIT Press.

Camacho, J. (2003). *The structure of coordination Conjunction and Agreement Phenomena in Spanish and Other Languages*. Dordrecht: Kluwer.

Chomsky, N. (1956). Three models for the description of language. In *IRE Transactions of Information Theory IT-2* (pp. 113-124).

Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control*, *2*, 137-167.

Chomsky, N. (1980). *Rules and representations*. New York: Columbia University Press.

Chomsky, N. (1995). *The minimalist program*. Cambridge. MA: MIT Press.

Chomsky, N. (2000). *New horizons in the study of language and mind*. Cambridge, UK: Cambridge University Press.

Chomsky, N. (2005). Three factor in language design, *Linguistic Inquiry*, *36*, 1-22.

Chomsky, N. (2007). Approaching UG from below. In U. Sauerland & H. M. Gärtner (*Eds.*), *Interfaces + Recursion = Language?* (pp. 1-30). Berlin: Mouton.

Chomsky, N. (2008). On phases. In R. Freidin, C. Otero, & M. L. Zubizarreta (*Eds.*), *Foundational issues in linguistic theory* (pp. 133-166). Cambridge. MA: MIT Press.

Chomsky, N. (2010). Some simple evo devo theses: How true might they be for language? In R. Larson, V. Déprez & H. Yamakido (*Eds.*), *The evolution of human language* (pp. 45-62). Cambridge: Cambridge University Press.

Chomsky, N. (2011). Language and other cognitive systems. What is special about language?. *Language Learning and Development, 7*, 263-278.

Chomsky, N. (2015). Some core contested concepts, *Journal of Psycholinguistic Research*, *44*, 91-104.

Corballis, M. C. (2007). Recursion, language and starlings, *Cognitive Science*, *31*, 69-704.

Corballis, M. C. (2011). *The recursive mind: The origins of human language, thought, and civilization*. Princeton, NJ: Princeton University Press.

Cutland, N. (1980). *Computability: an introduction to recursive function theory*. Cambridge: Cambridge University Press.

Dedekind, R. (1888). *Was sind was sollen die Zahlen?*, traducción José Ferreirós, 2014, Madrid: Alianza.

Fitch, T., Hauser, M.D. & Chomsky, N. (2005). The evolution of the language faculty: clarifications and implications. *Cognition*,97, 179-210.

Gentner, T. Q., Fenn, K. M., Margoliash, D., & Nusbaum, H. C. (2006). Recursive syntactic pattern learning by songbirds. *Nature*, 440, 1204–1207.

Gödel, K. (1934). On undecidable propositions of formal mathematical systems. In M. Davis (*Ed.*), *The undecidable* (pp. 39-74). New York: Raven Press.

Hauser, M., Chomsky, N. & Fitch, T. (2002). The faculty of language: What is, who has it, and how did it evolve?, *Science*, *298*, 1569-1579.

Jackendoff, R. & Pinker, S. (2005). The nature of the language faculty and its implications for evolution of language (reply to Fitch, Hauser, and Chomsky), *Cognition*, *97*, 211-225.

Jackendoff, R. (2009). Pararllels and nonparallels between language and music, *Music Perception*, *26*, 195-204.

Johannessen, J.B. (1998). *Coordination.* Oxford, UK: Oxford University Press.

Katz, J. & Pesetsky, D. (2009). The identity thesis for language and music. <http://ling.auf.net/lingBuzz/000959>.

Kinsella, A. (2010). Was recursion the key step in the evolution of the human language faculty? In H. van der Hulst (*Ed.*), *Recursion and human language* (pp. 179-191).

Kleene, S. C. (1938). On notation for ordinal numbers, *The Journal of Symbolic Logic*, *3*, 150-5.

Kleene, S. C. (1943). Recursive predicates and quantifiers, *Transactions of the American Mathematical Society*, *53*, 41-73.

Kleene, S. C. (1952). *Introduction to metamathematics*. Amsterdam: North-Holland Publishing.

Lewis, R.L., Vasishth, S. & Van Dyke, J.A. (2006). Computational principles of working memory in sentence comprehension. *Trends in Cognitive Sciences*, 10 (10), 447-454.

Monk, R. (1990). *Wittgenstein: The duty of genius*. New York: Free Press.

Moro, A. (2008). *The boundaries of Babel*. Cambridge, MA: MIT Press.

Mosterín, J. (2007). *Los lógicos*. Madrid: Austral.

Mota, S. (2013). La propiedad de la recursión en el "Tractatus Logico-Philosophicus" de Wittgenstein y su relación con la Teoría de la Computabilidad y la Lógica Matemática, 17, *Observaciones Filosóficas*.
http://www.obervacionesfilosoficas.net/lapropiedaddelarecursion.htm

Mota, S. (2014). La historia y la gramática de la recursión: una precisión desde la obra de Wittgenstein, *Pensamiento y Cultura*, *17*, 20-48.

Mota, S. (2015a). Sobre el concepto de recursión y sus usos, *Praxis Filosófica*, *40*, 153-181.

Mota, S. (2015b). ¿Por qué se usa 'recursión' cuando se quiere significar 'auto-inclusión'?: Clarificaciones conceptuales sobre la recursión en el programa chomskiano, *Revista de Lingüística Teórica y Aplicada*, 53, 171-191.

Odifreddi, P. (2001). Recursive functions: an archeological look. In C.S. Claude, M.J. Dinneen & S. Sburlan (*Eds.*), *Combinatorics, Computability and Logic* (pp. 13-31). London: Springer-Verlag.

Pinker, S. & Jackendoff, R. (2005). The faculty of language: What's special about it?, *Cognition*, *95*, 201-236.

Skolem, T. (1923). The foundations of elementary arithmetic established by means of the recursive mode of thought, without the use of apparent variables ranging over infinite domains. In J. Van Heijenoort (*Ed.*),

From Frege to Gödel. A source book in mathematical logic, 1879-1931 (pp. 302-333). Cambridge, MA: Harvard University Press.

Soare, R. (1996). Computability and recursion, *The Bulletin of Symbolic Logic*, *2*, 284-321.

Torretti, R. (1998). *El paraíso de Cantor. La tradición conjuntista en la filosofía matemática*. Santiago de Chile: Editorial Universitaria.

Vicari, G. & Adenzato, M. (2014). Is recursive language-specific? Evidence of recursive mechanisms in the structure of intentional action, *Consciousness and Cognition*, *26*, 169-188.

Wirth, N. (1986). *Algorithms and data structures*. Hemel Hempstead, UK: Prentice Hall. © N. Wirth 1985 (Oberon version: August 2004).

Wittgenstein, L. (1922). *Tractatus logico-philosophicus*. London: Routledge.

Wittgenstein, L. (1974). *Philosophical grammar*. Oxford: Blackwell.

Wittgenstein, L. (1975). *Philosophical remarks*. Oxford: Blackwell.

Wittgenstein, L. (1978). *Remarks on the foundations of mathematics*. Oxford: Blackwell.

Zhang, N. (2010). *Coordination in Syntax*. Cambridge, UK: Cambridge University Press.

2. Artículos

ARTÍCULO 1

Mota, S. (2015). Sobre el concepto de recursión y sus usos. Praxis Filosófica, 40, 153-181.

Praxis Filosófica
ISSN: 0120-4688
praxis@univalle.edu.co
Universidad del Valle
Colombia

Mota, Sergio
SOBRE EL CONCEPTO DE RECURSIÓN Y SUS USOS
Praxis Filosófica, núm. 40, enero-junio, 2015, pp. 153-181
Universidad del Valle
Cali, Colombia

SOBRE EL CONCEPTO DE RECURSIÓN Y SUS USOS

Sergio Mota
Universidad Autónoma de Madrid, España

Resumen

La noción de recursión se emplea en diferentes sentidos, adquiriendo una variedad de significados. Así, en unos casos se usa para caracterizar una regla esencial que constituye un modo de definición en un sistema. Este sentido tiene su origen en la Lógica Matemática y la Teoría de la Computabilidad. En otros casos se aplica para indicar la organización interna de una estructura, tal como sucede en la Ciencia Cognitiva y la Ciencia de la Computación, al tiempo que también se emplea en el sentido anterior en estas mismas disciplinas. El objetivo de este trabajo es mostrar en cada caso las diferencias en el uso de ambos sentidos.

Palabras clave: Recursión; regla gramatical; estructura; auto-inclusión; auto-referencia

Recibido: diciembre 17 de 2014 - Aprobado: abril 4 de 2015

Praxis Filosófica Nueva serie, No. 40, enero-junio 2015: 153 - 181

ISSN (I): 0120-4688 / ISSN (D): 2389-9387

On the concept of recursion and its uses

Abstract

The notion of recursion is commonly used in different ways, thus yielding a variety of meanings. In some cases it is used to characterize an essential rule in a system, which constitutes a mode of definition. This sense has its origin in Mathematical Logic and Computability Theory. In other cases it is applied to identify the internal configuration of a structure, as in Cognitive and Computer Science, where it is also used in the former sense. The goal of this paper is to clarify the differences between both uses or senses.

Keywords: Recursion; Grammatical Rule; Structure; Self-embedding; Self-reference

Sergio Mota. Doctorando en la Universidad Autónoma de Madrid, España. Licenciado en Psicología y Máster en Crítica y Argumentación Filosófica por la misma Universidad. Sus áreas de investigación son: el papel de la recursión en la obra de Wittgenstein. La recursión en la Ciencia Cognitiva del Lenguaje. Revisión de la Biolingüística chomskiana. Filosofía de la Lógica y de la Matemática. Filosofía de la Ciencia y del Lenguaje. Filosofía de la Mente.

Dirección posta: Departamento de Psicología Básica, Universidad Autónoma de Madrid, Campus de Cantoblanco, CP: 28049 Madrid, España.

Dirección electrónica: sergio.mota.v@gmail.com

SOBRE EL CONCEPTO DE RECURSIÓN Y SUS USOS*

Sergio Mota
Universidad Autónoma de Madrid, España

El sentido original de 'recursión'

El sentido original de recursión está claramente establecido dentro de la teoría de la computabilidad y de la lógica matemática (Soare, 1996), si bien dentro de estos dominios parece haberse empleado en dos sentidos diferentes. Uno de ellos se refiere a la definición por recursión (o definición recursiva) y el otro a la noción de computabilidad (y lo mismo vale para los adjetivos 'recursivo' y 'computable'; véase, por ejemplo, Soare, 2009).

El primero de los sentidos mencionados, el de la definición por recursión, hace referencia a un método usado para definir funciones y predicados. Su origen se remonta al siglo XIX y fue ya empleada por autores como Dedekind o Peano (ver Soare, 1996, 2009 para más detalles y referencias).

Así, una función (o predicado) está definida, en un sentido técnico, por recursión cuando para definirla para un argumento *y* hacemos uso de sus propios valores previamente computados para argumentos menores que *y*; pudiendo emplearse también funciones previamente definidas (Soare, 1996, 2009; véase también Gödel, 1931; Kleene, 1952; Cutland, 1980).[1]

* Mi gratitud a José Manuel Igoa y David J. Lobina por su lectura atenta y valiosos comentarios a una versión previa. Asimismo, mi agradecimiento a los revisores anónimos de *Praxis Filosófica*, cuyas observaciones han ayudado a mejorar el texto.

[1] Esta definición supera algunas imprecisiones en Mota (2013, 2014); aunque bien es cierto que en tales trabajos se sigue correctamente a Cutland (1980, p. 32): una función *f* es recursiva; esto es, está definida por recursión, si cada (nuevo) valor (de *f*) se especifica/define usando sus propios valores previamente definidos (computados), siguiendo, obviamente, la regla recursiva que corresponda.

Este es el sentido en el que Dedekind y Peano usaron dicho concepto y que constituye su significado original (Soare, 1996, pp. 286-287). En este sentido, Epstein y Carnielli (1989) indican que en su forma más simple, la definición recursiva de una función *f* sería como sigue (siendo *g* una función previamente definida): $f(0) = m$; $f(n+1) = g(f(n))$.[2]

Por otro lado, conviene no confundir la definición inductiva, la cual sirve para definir términos como 'número natural', 'objeto', etc., con una definición por recursión (o 'recursiva'). Sólo la segunda, y no la primera, es la empleada en la Lógica Matemática y la Teoría de la Computabilidad para definir funciones y/o predicados (Kleene, 1952, p. 217). Por consiguiente, definir un término mediante una definición inductiva no convierte al objeto definido en recursivo (ver infra). La recursión es una propiedad de la definición, de la regla (Mota, 2013, 2014).

Así, desde la lógica matemática, la matemática y la ciencia de la computación, podemos distinguir entre las definiciones inductivas y co-inductivas (que proporcionan los principios duales a los de las primeras).[3] Las primeras proporcionan, o están en la base de, dos principios diferentes que, *a priori*, no hay que confundir, los principios de inducción (que permite probar ciertas propiedades internas de los objetos tratados, por ejemplo de los números naturales o de las fórmulas proposicionales), y de recursión (que, como se verá, garantiza la buena definición de las funciones recursivas), las segundas proporcionan los principios de co-inducción (que también permiten probar ciertas propiedades internas de los objetos tratados) y de co-recursión (que garantizan la buena definición de funciones a través de procesos). Desde un punto de vista formal, estas definiciones se pueden contextualizar de varias maneras: en la Teoría de Conjuntos, en la Teoría de categorías, entre otras. Sin embargo, no es el objetivo de este trabajo entrar en cada una de estas aplicaciones.

Para una adecuada contextualización de la noción de recursión, creo conveniente precisar la distinción entre 'definición inductiva' y 'definición recursiva' más detalladamente. Así, en el marco particular de una teoría axiomática de conjuntos, por ejemplo, Zermelo-Fraenkel (ZF) con el axioma de elección, se puede formalizar la noción de Conjunto Inductivo. Un conjunto I se dice inductivo si (a) el conjunto vacío '∅' pertenece a I y

[2] Así, las funciones no son intrínsecamente recursivas, sino que lo son en virtud de cómo se definen. Como ya he señalado en otro trabajo (Mota, 2014, p. 22) una definición recursiva tal es una regla gramatical en el sentido establecido por Wittgenstein (*Cf.* Mota, 2013, nota 3).

[3] Agradezco a los revisores anónimos de *Praxis Filosófica* sus comentarios y sugerencias en este punto, los cuales e intentado seguir con toda fidelidad y precisión.

(b) siempre que un conjunto X pertenece a I, entonces también el sucesor S(X), obtenido de la unión de X con el conjunto que tiene como único elemento a X, esto es, $S(X) = X \cup \{X\}$. Así, pueden definirse cómodamente el conjunto de los números naturales. Esta definición parece estar relacionada con la concepción iterativa de un conjunto (Boolos, 1971). La definición inductiva arriba presentada proporciona de inmediato el llamado Principio de Inducción (matemática), el cual permite mostrar propiedades internas de los objetos así construidos (Boolos, *Op. cit.*, pp. 223-224). Así, hay ciertamente una analogía entre la definición inductiva de los números naturales con la definición inductiva aplicada a la formación de conjuntos (op.cit., p. 223), pero no hay que confundir la definición inductiva con el Principio de Recursión, que garantiza la buena definición de distintas operaciones como la suma o el producto (Hrbacek y Jech, 1999). Siguiendo a Hrbacek y Jech (1999, p. 46 y ss.), el Principio de Recursión permite definir funciones sobre N. Por ejemplo, una de tales funciones es la adición (ver nota 6). Una interesante distinción que estos autores hacen (*Ibid*.) es entre la definición recursiva de tales funciones y la prueba por inducción que sirve para probar ciertas propiedades de la adición, como la conmutativa. En este caso, para $m, n \in N$, se prueba que $m+n = n+m$, para después probar, mediante inducción sobre n, que $m+(n+1) = (n+1)+$m. finalmente se prueba, mediante inducción sobre m, que $(m+1)+(n+1) = (n+1)+(m+1)$.[4]

Por otro lado, una forma de expresar la inducción matemática es como sigue (véase Boolos, 1971 para otras versiones):[5]

$$(P)\,[P0 \,\&\, (n)\,[Pn \rightarrow PSn] \rightarrow (n)Pn]$$

Las distintas clases de funciones recursivas hacen uso, por tanto, del Principio de Recursión, como muestra claramente Kleene (1943), mediante el esquema V. Tales clases diferentes de funciones así definidas son: las primitivas, en las que la inducción se aplica sobre una variable (véase Skolem, 1923; Gödel, 1931);[6]

[4] El principio de inducción matemática es, por tanto, diferente al de recursión. Por otro lado, una regla de sustitución como (a) 3+3 = ((2+2)+1)+1 es diferente a una regla recursiva como (b) 3+3 = (3+2)+1 = ((3+1)+1)+1. La segunda posibilita el paso recursivo de '3+3' a '(2+3)+1' y, sólo desde de ahí, a '((2+2)+1)+1'. Sin embargo, conviene advertir, una cosa es el paso desde '3+3' a '(2+3)+1' (permitido por la regla recursiva), o desde '(2+3)+1' a '((2+2)+1)+1' (también permitido, aunque cambiando el parámetro) y otra cosa es el paso desde '3+3' a '((2+2)+1)+1', el cual no es un paso recursivo directo; es decir, (a) no instancia directamente a '$a+(b+1) = (a+b)+1$'. Por otro lado, una prueba por inducción matemática también puede hacer uso de la iteración (véase Wittgenstein, 1974, 32).

[5] En palabras: si 0 tiene la propiedad *P* y si para todo n que tiene *P* su sucesor también, entonces todo *n* tiene *P*.

[6] Un ejemplo de función recursiva primitiva es el de la suma: $a+0 = a/a+1 = a'$ Def., [caso base]; $a+(b+1) = (a+b)+1$ Def. [paso recursivo].

las generales, en las que se aplica sobre, al menos, dos variables simultáneamente y que incluye a las primeras (véase Gödel, 1934);[7] y las parciales que incluyen a las anteriores como aquellas funciones parciales para las cuales está definido todo el conjunto de sus argumentos (esto es, como funciones recursivas totales) y que reciben su nombre precisamente porque una función no necesita estar definida para todas las n-tuplas de números naturales que toma como argumentos (Kleene, 1938, 1943, 1952).[8] Tal clase de funciones recursivas identifica correctamente la clase de las funciones computables (Kleene, 1952).[9]

Otros formalismos propuestos dentro de la teoría de la computabilidad y la lógica matemática también hacen uso de la definición recursiva. Tal es el caso del formalismo de Post (1921, 1943, 1944). Post formalizó un sistema capaz de generar todas las proposiciones de la lógica de enunciados (alias "tautologías", *Cf.* Wittgenstein, 6.1). Así, sea g un enunciado de la lógica proposicional y P una variable operacional aplicada a dicho enunciado, el sistema produce un enunciado g' que sustituye a g. Esto se expresa en su sistema de producción normal como sigue: *gP→Pg'* (Post, 1943, p. 199). De esta manera, el método de decisión formulado por Post, similar al proporcionado por Wittgenstein (1922), permite decidir en un número finito de pasos si una fórmula es o no una tautología (esto es, una proposición

[7] Un ejemplo de función recursiva general es: $\varphi(0,y) = \psi(y)$ Def., $\varphi(x+1, 0) = \chi(x)$ Def., $\varphi(x+1, y+1) = \varphi(x, \varphi(x+1, y))$ Def. Aunque en '$\varphi(x, \varphi(x+1, y))$' las funciones interna y externa son denotadas por 'φ' ambas son funciones diferentes, de primer orden la interna de y de segundo orden la externa (ver infra).

[8] Una función parcial es: $f(x) = f(x-1)/2$. Esta función sólo está definida para los números naturales impares tomados como argumentos (*Cf.* Boolos y Jeffrey, 1974; para una definición más técnica véase Kleene, 1938, 1943, 1952).

[9] Como Soare (2009) y otros autores señalan (véase Sieg, 1997), Church cometió un error al identificar las funciones recursivas generales con las computables. Este error, por un lado, y la identificación extensional de las funciones recursivas parciales con la máquina de Turing (esto es, que ambos formalismos computan la (misma) clase de las funciones computables o calculables; véase Epstein y Carnielli, 1989), me llevaron a definir explícitamente la Tesis de Kleene y la Tesis de Turing-Kleene, que se pueden condensar en la siguiente formulación: *una función es computable si y sólo si es una función recursiva parcial (o está definida por el formalismo de Kleene)* –que hace referencia a la Tesis de Kleene– *o, de forma equivalente, si es computable/calculable por una máquina de Turing* –que junto con lo anterior constituye la Tesis de Turing-Kleene (Mota, 2013, 2014). La *Tesis de Turing-Kleene* bien podría substituir a la *Tesis de Church-Turing* dado que, como señalo, fue Kleene, y no Church, quien identificó correctamente la clase de las funciones computables (Mota, 2014). Por otro lado, se usa la Tesis de Church para hacer alusión a la equivalencia extensional de las funciones recursivas generales definidas por Gödel, el cálculo-λ de Church y las máquinas de Turing, todas ellas propuestas para caracterizar la noción de algoritmo (*Cf.* Soare, 1996).

de la lógica (proposicional)).[10] Su tesis, conocida como la Tesis de Post, establece que un conjunto no vacío (como el de las proposiciones de las lógica de enunciados) es efectivamente numerable si y sólo si es derivado de un sistema de producción (normal) [derivado de su sistema canónico (normal)] (Soare, 2009).[11] Un sistema tal hace uso de la definición por recursión, tal como Post (1943, p. 201) indica (él hace uso de la expresión definición por inducción pero se pueden intercambiar), y como he definido en otro trabajo (Mota, 2013; 2014, p. 31):

$P^{(0)}$'$(g) = g$ Def.,
$P^{(n)}$'$(g) = P$'$P^{(n-1)}$'(g) Def.

Como señalé al comienzo de este apartado, el concepto de recursión se ha aplicado con cierta ambigüedad dentro de la teoría de la computabilidad. Así, tal y como señala Soare (1996, 2009), dicha noción se comenzó a aplicar para significar 'computabilidad'. Sin embargo, el término 'computabilidad' (o 'efectivamente computable') se refiere a una función para la que existe un procedimiento (efectivo) mecánico finito (esto es, un procedimiento generativo/computacional o algoritmo) por medio del cual es computada o calculada; esto es, si se ha definido para ella una secuencia de reglas con las que, dado un *input*, se obtuviera un valor o *output*, a través de una secuencia finita de operaciones. Desde 1996, Soare ha propuesto volver a emplear el concepto de recursión en su sentido original, distinguiéndolo del de computabilidad, un concepto relacionado pero no coextensivo, dado que hay formalismos que no proceden recursivamente, como la máquina de Turing (véase Turing, 1937; para una definición esquemática ver infra).

[10] El llamado 'problema de la decisión' (*Entscheidungsproblem*), formulado por Hilbert a comienzos del Siglo XX, consistía en establecer un procedimiento de decisión (*Entscheidungsverfahren*) que permitiera determinar o decidir, en un número finito de operaciones, si una expresión o fórmula bien formada es o no válida. Obviamente, esto se relaciona con su *programa finitista*, por medio del cual pretendía probar la consistencia de la aritmética. Dicho problema fue mostrado irresoluble por Gödel (1931), Church (1936), Turing (1937) y Post (1943).

[11] Post (1944) también se centró en los conjuntos recursivamente numerables, distinguiendo entre éstos y los conjuntos recursivos. Un conjunto S es recursivo si se puede establecer un método efectivo –un procedimiento mecánico finito– para determinar si, por ejemplo, un número entero positivo n pertenece o no al conjunto S. Un conjunto S es recesivamente numerable si existe un procedimiento mecánico efectivo para numerar efectivamente los elementos de S (adelanto aquí que tales procedimientos pueden ser las funciones recursivas primitivas, generales y parciales; véase infra para más detalles). Tal y como señala Soare (1996, 1999, 2009) 'recursivamente numerable' significa 'efectivamente numerable' y 'conjunto recursivo' significa 'conjunto efectivo'.

En todo caso, aunque la máquina de Turing proceda iterativamente, a diferencia de las funciones recursivas, esto no quiere decir que no se pueda establecer una regla recursiva que defina lo que haga. Así, tal y como he indicado en otro trabajo (Mota, 2014), haciendo uso del siguiente diagrama de estados (simplificado):

… 0 0 0 0 … (esta es la cinta simplificada)

a. La primera instrucción es: E0, 0, E1, 1: la máquina, estando en E0 escaneando '0', pasa al estado E1 substituyendo '0' por '1': … 1 0 0 0 … [E1 se define como $\tilde{O}$ sobre 0]
b. En este segundo paso, las instrucciones son: E1, 1, E2, >: la máquina, estando en E1 escaneando '1', pasa a E2 moviéndose a la derecha: *[*E2 se define como $\tilde{O}$ sobre E1 o como $\tilde{O}$ sobre $\tilde{O}(0)$*]*
c. En el tercer paso, las instrucciones son: E2, 0, E3, 1: la máquina, estando en E2, escaneando '0', pasa a E3 substituyendo '0' por '1': … 1 1 0 0 …*[*E3 se define como $\tilde{O}$ sobre E2, o como $\tilde{O}$ sobre $\tilde{O}'\tilde{O}(0)$*]*

(y así sucesivamente hasta que la máquina se detiene; omito aquí ese paso)

Es posible establecer el siguiente sistema de ecuaciones recursivas (Mota, 2014, pp. 31-32):

$\tilde{O}^{(0)}'(0) = 0$ Def.,
$\tilde{O}^{(n)}'(0) = \tilde{O}'\tilde{O}^{(n-1)}'(0)$ Def.

La primera ecuación expresa el estado E0 de la máquina, mientras que la segunda ecuación expresa la definición recursiva de los distintos términos/estados. Es importante señalar que '$\tilde{O}$' es una variable operacional cuyos valores son las distintas operaciones que realiza la máquina de Turing. Esto sólo indica que la serie generada por una máquina de Turing puede definirse recursivamente en el nivel de análisis relacionado con *qué* hace tal procedimiento. Sin embargo, respecto a *cómo* procede u opera es claro que lo hace iterativamente, esto es, pasando en sucesión de un estado a otro.

Así, aunque, por ejemplo, la función suma se defina recursivamente, no tiene que proceder (operar o implementarse, tanto de manera abstracta como en tiempo real) necesariamente de manera recursiva (*Cf.* Abelson et al., 1996). Una implementación recursiva instancia una definición recursiva, y un ejemplo del primer caso sería el siguiente: si se quiere computar 2+2, el procedimiento invoca un valor previamente computado para un argumento menor, esto es, se computaría (2+1)+1; con lo que obtenemos '2+2 = (2+1)+1 = ((2+0)+1)+1'; en cambio, una implementación iterativa de '2+2' se puede definir del siguiente modo: primero se obtiene el sucesor de 1 (1+1) y al

(último) resultado (2) se le añade 1, y al resultado (3) se le añade 1, hasta que se alcanza el valor '4' (lo que se puede expresar como 2+2 = 1+1+1+1). Mediante tal implementación iterativa, una operación genera sucesivamente valores desde un argumento a otro sin usar valores previamente calculados para argumentos menores.

En cualquier caso, teniendo en cuenta que un sistema formal como los sistemas que acabo de mencionar es una serie de formas, esto es, cada uno de sus términos están ordenados por relaciones internas (Wittgenstein, 1975, § 154) y que una serie de formas queda definida por [*a, x, O'x*] (Wittgenstein, 1922, 5. 2522)[12] o por $O^{(0)}{}'(a) = a$ Def.; $O^{(n)}{}'(a) = O'O^{(n-1)}{}'(a)$ Def. (Mota, 2013, 2014) podemos definir la forma general de un procedimiento mecánico finito como [$\tilde{\sigma}$, $\tilde{\varepsilon}$, $\tilde{O}(\tilde{\varepsilon})$] (Mota, 2013, §2; 2014, p. 30) o como $\tilde{O}^{(0)}{}'(\tilde{\varepsilon}) = \tilde{\varepsilon}$ Def.; $\tilde{O}^{(n)}{}'(\tilde{\varepsilon}) = \tilde{O}'\tilde{O}^{(n-1)}{}'(\tilde{\varepsilon})$ Def. (Mota, *Ibid.*). Así, bajo esta formulación, que expresa una variable, caen los siguientes ejemplos:

a. En el caso de una función recursiva,
   $\tilde{\varepsilon}, \tilde{O}\tilde{\varepsilon}, \tilde{O}\tilde{O}\tilde{\varepsilon},\ldots$
   2+2, 2+2+1, 2+2+1+1,…
   (2+2), (2+3), (2+4),…
   se puede instanciar como '2+4 = (2+3)+1'.

b. En el caso del formalismo de Turing,[13]
   $\tilde{\varepsilon}, \tilde{O}\tilde{\varepsilon}, \tilde{O}\tilde{O}\tilde{\varepsilon},\ldots$
   $\tilde{\varepsilon}, \tilde{O}\tilde{\varepsilon}, \tilde{O}\tilde{O}\tilde{\varepsilon},\ldots$
   E0, $\tilde{O}(E_0)$, $\tilde{O}\tilde{O}(E_0)$,…

que en el caso de la generación de los números naturales se puede escribir:

[12] En 5.2522, Wittgenstein explica en qué consiste esta notación. Así, "[e]l primer término…es el comienzo de la serie de formas, el segundo es la forma de un término x cualquiera de la serie y el tercero la forma de aquel término de la serie que sigue inmediatamente a *x*".

[13] Bien es cierto que el formalismo de Turing es considerado como un modelo de computabilidad, pero esto no niega que se pueda concebir como una adecuada formulación de un sistema formal o, también, de un procedimiento mecánico finito (véase Gödel, 1964, pp. 71-72). Por otro lado, los sistemas formales considerados aquí pueden entenderse como objetos matemáticos formales (*Cf.* Soare, 1996), considerados, bajo la perspectiva del formalismo; esto es, bajo una concepción no-descriptivista de la matemática. En otro sitios (Mota, 2013, 2014), y siguiendo la concepción formalista y wittgensteiniana de la matemática (y de la lógica), he considerado que tales objetos matemáticos o sistemas formales no son sino construcciones o invenciones constituidas por reglas gramaticales, las cuales no son descripciones de ninguna realidad (empírica o platónica).

E_0; Sub0/1' Esc0' (E_0); >' Esc1' (Sub0/1' Esc0' (E_0)),[14] … que significa:

E_0, E1 (=Sub0/1' Esc0' (E_0)), E_2 (= >' Esc1' (Sub0/1' Esc0' (E_0))),…y así sucesivamente.[15]

Por tanto, cada estado/configuración o término se puede definir como sigue:

$\tilde{O}^{(n)}'(\tilde{\varepsilon}) = \tilde{O}'\tilde{O}^{(n-1)}'(\tilde{\varepsilon})$.

Así, la expresión 'Sub0/1' Esc0'>' Esc1' (Sub0/1' Esc0' (E_0))' muestra que, estando en E0, se aplican las operaciones 'Esc0' y 'Sub0/1', lo que permite calcular '1', pasando así al estado E1. Mediante la aplicación de las operaciones 'Esc1', '>', 'Esc0' y 'Sub0/1' la máquina calcula '2' y así sucesivamente. La recursión caracteriza la definición de los *n* estados/configuraciones, términos o formas (v.gr., '$\tilde{O}^{(n)}'(\tilde{\varepsilon})$') efectivamente computados por una máquina de Turing. Esto caracteriza *qué* hace una máquina de Turing (lo que se puede aplicar al resto de formalismos), esto es, generar recursivamente una serie de términos, pero *cómo* lo hace queda descrito por medio de la iteración, esto es, por medio de la repetición sucesiva de operaciones sobre el último resultado calculado (*Cf.* Mota, 2013, 2014).

El análisis que acabo de ofrecer me permite hacer la siguiente observación. Tal como indica Soare (2009), puede establecerse una diferencia intensional entre *sistemas generacionales*, que proporcionan un procedimiento o algoritmo para listar un conjunto de, digamos, números naturales, como los sistemas de producción de Post, y *sistemas computacionales*, los cuales computan funciones, como las funciones recursivas primitivas, generales, parciales o las funciones computables por una máquina de Turing. Es indudable que tales diferencias intensionales no son esenciales, lo cual no quiere decir que sean irrelevantes, para la formulación de la forma general de un procedimiento mecánico finito. De hecho, todos ellos caen bajo esa misma forma general.[16] No obstante, podemos encontrar *sistemas computacionales* en el sentido que acabo de señalar que listen un conjunto de números naturales; esto es, que sean *sistemas generacionales*. De tal manera que una función recursiva también proporciona un procedimiento generativo (o generacional), por medio del

[14] 'EscX' significa 'escanear X'; 'SubX/Y' significa 'sustituir X por Y'; '>' significa 'mover a la derecha'.

[15] Por ejemplo, E1 es el estado de la máquina obtenido estando previamente en E0, escaneando 0, y substituyendo 0 por 1.

[16] Tal forma general muestra lo que todos ellos tienen en común como procedimientos mecánicos finitos, algoritmos o sistemas matemáticos formales: un conjunto de términos que generan (u operan sobre) otros términos o secuencia de ellos mediante la aplicación de operaciones siguiendo un conjunto de reglas (*Cf.* Gödel, 1934).

cual se puede listar (o generar) un conjunto de números naturales (*Cf.* Post, 1944).[17] Pero también es posible encontrar el caso inverso, es decir, un sistema de producción *gP→Pg'* donde g puede estar por una cadena de '1s' y 'bs', que a su vez están por un número natural n, cadena a la que al aplicar una variable operacional *P* el sistema produce otra secuencia de '1s' y '*bs*' tal que genera g'; esto es, otra secuencia. Esto se puede representar como *b*1*bP→P*1*bb*1 (Post, 1944, p. 307). De este modo, cada número natural *n* se puede definir como una función de *n* aplicaciones de *P* sobre *g*: $P^{(n)}$'(*g*) = P'$P^{(n-1)}$'(*g*) Def.

Otro ejemplo de esto puede mostrarse con respecto al cálculo-λ definido por Church (1932, 1936). Así, Church, (1936, p. 91), define los números naturales de la siguiente manera:[18]

$1 \rightarrow \lambda ab.\ a(b)$
$2 \rightarrow \lambda ab.\ a(a(b))$
$3 \rightarrow \lambda ab.\ a(a(a(b)))$

Tal formulación puede convertirse en términos de funciones (*Cf.* Frascolla, 1994; Marion, 1995, 1998; Odifreddi, 2001). Así, como ya he señalado en otro lugares (Mota, 2013, 2014), donde también subrayé el parecido de esta formulación con la de Wittgenstein unos años antes (1922), podemos transformar ese sistema en el sistema de ecuaciones que ofrezco a continuación, en donde la definición recursiva para ambos procedimientos es como sigue: Ω^0'$x = x$ (en el caso del formalismo de Church: $F^{(0)}(X) = X$) [*el caso base*]; $\Omega^{(v+1)}$'$x = \Omega$'Ω^v'x (en el caso del formalismo de Church: $F^{(n+1)}(X) = F(F^{(n)}(X))$) [*el paso recursivo*].

En suma, la recursión, que se diferencia de la iteración conceptualmente aunque ambas puedan coexistir en diferentes dominios de análisis, es una propiedad de una definición o una regla que indica que para definir un valor se hace uso de un valor o valores previamente computado/s para un argumento (o varios) menor/es (*Cf.* Cutland, 1980; Odifreddi, 2001). Esto es así

[17] De hecho, el propio Post (1944, pp. 307-308) reconoce que el formalismo de las funciones recursivas parciales amplió el concepto de conjunto recursivamente numerable. En todo caso, en ese mismo trabajo (p. 306), indica que un conjunto de enteros positivos (los números naturales) es recursivamente numerable si hay una función recursiva $f(x)$ cuyos valores, para enteros positivos como valores de x, constituye tal conjunto (esto es, el rango de $f(x)$). Si tal función recursiva permite generar los números naturales en su orden natural, entonces también se ha establecido un procedimiento para determinar si un entero natural pertenece o no a tal conjunto; de ahí se desprende, de acuerdo con las definiciones dadas más arriba, que tal conjunto es, también, recursivo (*Cf.* Post, 1944, p. 311).

[18] Estas expresiones tienen la forma $\alpha \rightarrow A$, la cual ha de leerse como "α representa a/está por A" (Church, 1936, p.91). En este sentido α puede 'rescribirse' como A.

independientemente de que la función se defina para todos los argumentos, pues como muestra Kleene (1943), todas las clases de funciones recursivas hacen uso de la definición por recursión, expresada por medio del esquema V (*Cf.* Soare, 1996, p. 287). En este punto se equivoca Tomalin (2007, 2011), cuando trata de precisar la noción de recursión dentro de la teoría sintáctica, pues él apunta que la noción de recursión se define de manera distinta en cada clase de funciones recursivas. El principio de recursión, además de garantizar la buena definición de las funciones recursivas, permite que manejemos un único uso de 'recursión', congruente, por tanto, con la definición dada por Soare e indicada unas líneas más arriba.

A continuación mostraré cómo se emplea la noción de recursión en la Ciencia Cognitiva, para después mostrar los paralelismos y las diferencias en el tratamiento que se hace de este concepto en la Ciencia de la Computación.

Recursión Vs. auto-inclusión en la Ciencia Cognitiva

En este apartado me propongo analizar conceptualmente las relaciones entre 'recursión' y 'auto-inclusión', empezando por la noción de 'recursión' aplicada a las estructuras, para después analizar la relación con los mecanismos recursivos empleados para generar tales estructuras.

En términos generales, se entiende que una estructura recursiva es aquella que contiene un constituyente A (o estructura) dentro de otro constituyente B (o estructura) del mismo tipo (véase por ejemplo Pinker y Jackendoff, 2005; Jackendoff y Pinker, 2005; Everett, 2005, 2009; Corballis, 2007, 2011; Moro, 2008; Karlsson, 2010; Kinsella, 2010; Zwart, 2011; Jackendoff, 2011; Lobina, 2011, 2014a; 2014b; Arsenijević y Hinzen, 2010, 2012).

En esta definición entran diferentes tipos de estructuras, ya sea en un nivel de descripción concreto, como las que contienen dos sintagmas nominales o dos oraciones de relativo (Karlsson, 2010, p. 51) o en un nivel mayor de abstracción, como los esquemas del tipo [especificador-[núcleo-complemento] incluidos dentro de otros esquemas idénticos, dado que tanto el especificador como el complemento pueden incluir tales esquemas: […*Núcleo*…-[*Núcleo*-[…*Núcleo*…]]] (Moro, 2008).

Luuk y Luuk (2011) definen las estructuras con auto-inclusión como acabo de hacer, pero ellos se muestran más críticos con la aplicación de la noción de recursión a tales estructuras, como parece que Chomsky (1965) hiciera al definir las estructuras con auto-inclusión. Así, una estructura con auto-inclusión es aquella en la que el sintagma (o estructura) A esta auto-incluido en B si A está anidado en B y, además, A es un sintagma (o una estructura) del mismo tipo que B.

Desde una perspectiva histórica, tales estructuras se han venido relacionando con reglas de rescritura del tipo *S→aSb*. Sin embargo, como muy bien han señalado Luuk y Luuk (2011), una regla tal puede generar tanto secuencias que exhiben auto-inclusión como secuencias que no la exhiben. En concreto, estos autores han puesto de relieve que una regla como AB→AABB, que genera secuencias como *aabb*, *aaabbb*,…, no entraña necesariamente auto-inclusión, y a la inversa, secuencias como estas, incluso admitiendo que presentan auto-inclusión en su estructura interna, no tienen por qué haberse generado necesariamente mediante reglas recursivas. En esta misma línea, Lobina (2014a, 2014b) ha mostrado que una secuencia de reglas como *A*→aB, B→*aC*, *C*→*bD*, *D*→*b* genera secuencias como aabb, las cuales, como en el caso anterior, pueden o no exhibir auto-inclusión.

Lo que quiero subrayar aquí es que aunque AB→AABB sea una regla recursiva, de acuerdo con la definición original del término expuesta en el apartado anterior, la secuencia *aaabbb*, resultado de su aplicación, no puede juzgarse como recursiva o con auto-inclusión. En otras palabras, las estructuras subyacentes a las secuencias de este tipo pueden ser o no "recursivas" con independencia de la propiedad recursiva de la regla.

Esta falta de precisión conceptual se muestra de forma palmaria en Jackendoff (2011), quien distingue entre dos tipos de recursión, la *recursión formal* y la *recursión estructural*. Jackendoff aplica la primera a las reglas y afirma que un conjunto de reglas es recursivo si éstas pueden aplicarse a su propio resultado un número ilimitado de veces a partir de un conjunto de ítems primitivos o no definidos (*Op. cit.*, p. 591). Esta definición, sin embargo, es incorrecta, ya que no distingue la recursión de la iteración. Por otra parte, define la recursión estructural en los términos arriba señalados, afirmando que una estructura recursiva es aquella que incluye constituyentes dentro de otros (del mismo tipo) con una profundidad ilimitada (*Op. cit.*, p. 592). Después de establecer tal distinción, Jackendoff sostiene que de estos dos usos del término, el de recursión estructural es más útil para los propósitos de la Ciencia Cognitiva. Así, en el lenguaje natural, nos dice Jackendoff, tenemos que establecer primero su dominio de recursión estructural y, si podemos hacerlo, entonces podemos concluir que las reglas que definen el dominio son formalmente recursivas. En otras palabras, según Jackendoff, podemos inferir la recursión formal a partir de la recursión estructural. O dicho de una manera ligeramente diferente, que la *única* evidencia de la recursión formal es la recursión estructural.

Ciertamente, el análisis que presenta Lobina (2014b) en términos de reglas no recursivas (ver supra), las cuales pueden generar secuencias tanto sin como con auto-inclusión, es válido para poner en duda la aseveración

de Jackendoff. Sin embargo, también se puede desmontar la aseveración de Jackendoff sin acudir al argumento de las reglas no recursivas. De hecho, tal y como he mostrado en otro trabajo (Mota, 2014), se puede definir una regla recursiva sin necesidad de establecer primero un dominio de auto-inclusión. Así, una regla recursiva como '$f(x) = s(f(x\text{-}1))$', que puede perfectamente definir una función gramatical, expresa una definición recursiva en la cual no hay rastro de auto-inclusión. Esta regla define valores haciendo uso de valores previamente computados para argumentos menores, estableciendo una relación de intercambio o substitución entre expresiones, pero nunca de auto-inclusión (Mota, 2013, 2014). Lo recursivo es, por tanto, la definición, la regla que subyace a dicha ecuación matemática y la constituye, pero ninguna de las expresiones por separado lo es (Mota, 2014, p. 41).

Un asunto no menos importante es el hecho de que una regla recursiva no tiene por qué ser necesariamente una regla de rescritura. *Merge*, el actual procedimiento generativo/computacional que subyace a la facultad del lenguaje dentro del *Programa Minimista* (Chomsky, 1995; Hauser, Chomsky y Fitch, 2002), puede definirse en términos de un sistema de ecuaciones recursivas y no en términos de reglas de rescritura y no por ello deja de ser recursivo.

En definitiva, que una regla recursiva no pueda caracterizarse mediante la propiedad de la auto-inclusión dado que la relación entre expresiones es de substitución o intercambio y no de (auto-)inclusión (Mota, 2013, 2014), muestra claramente que la auto-inclusión no puede ser razón a partir de la cual inferir o deducir la propiedad de la recursión. Por ello, si usamos 'recursión' en su sentido original, la auto-inclusión no justifica la propiedad de la recursión, pues ambas propiedades son independientes, y la razón por la que se predica recursión (en tal sentido original) no es, en ningún caso, la auto-inclusión (Mota, 2014).

La misma línea argumental puede seguirse para Merge. Dentro del actual *Programa Minimista*, Chomsky (1995, 2007a) establece que un lenguaje-I es un procedimiento generativo/computacional que genera recursivamente una ordenación (o serie) infinita de expresiones jerárquicamente estructuradas mediante la operación denominada "*Merge* (o ensamble)".[19] Lo que hace un procedimiento mecánico finito de este tipo es construir recursivamente objetos sintácticos (Chomsky, 1995, p. 226). Como es bien sabido, las

[19] Para Chomsky (1959, 1995, 2005, 2007, 2008, 2010, 2011, 2012), el Lenguaje-I es concebido como un procedimiento generativo o sistema de cómputo capaz de generar una cantidad potencialmente infinita de expresiones jerárquicas internas. De este modo, la teoría de la gramática, que comprende el estudio del lenguaje así entendido, es el estudio de una clase especial de funciones recursivas dentro de la teoría de la computación (o computabilidad) (Chomsky, 1959, 2012).

expresiones u objetos sintácticos generados por *Merge* son interpretados por dos sistemas, a saber: el sensorio-motor y el conceptual-intencional (véase para más detalles Hauser Chomsky y Fitch, 2002; Chomsky, 2006, 2007a, 2007b, 2008, 2010, 2011).

Tal y como he defendido en diferentes lugares (Mota, 2013, 2014), Chomsky ha empleado correctamente la recursión como propiedad que define su procedimiento generativo/computacional. En esos mismos trabajos defino recursivamente de forma clara tal procedimiento; lo cual muestra qué hace *Merge*, a saber, generar recursivamente objetos sintácticos (Chomsky, 1995). Así, la forma general de *Merge* se puede establecer de la siguiente manera: [*N*, *Os*, $M(O_s)$]. *N* hace referencia a las piezas léxicas (o ítems léxicos) y objetos sintácticos que configuran la numeración (o repertorio seleccionado de objetos que entran en una derivación sintáctica), O_s hace referencia a un objeto sintáctico cualquiera de la serie generada (que puede tomar n-valores: si $n = 1$, entonces *Os* = X; si $n = 2$, entonces *Os* = X, Y, y así sucesivamente) y $M(O_s)$ hace referencia a un objeto sintáctico nuevo generado a partir de O_s mediante la aplicación de *M* ( ) –esto es, *Merge*. Este procedimiento genera/define recursivamente objetos sintácticos, independientemente de cuál sea su organización interna a efectos de la auto-inclusión de constituyentes (o esquemas) (Mota, 2013, nota 2; 2014, p. 38).

Lo que hace *Merge* (entendida aquí como una operación binaria) es tomar dos ítems léxicos, X,Y, para crear con esa unión un nuevo objeto sintáctico Z (X,Y = {X,Y} = Z). Mediante otra aplicación, Merge genera un nuevo objeto sintáctico, formado a partir de un objeto previamente creado. Así, y de manera esquemática, vemos que lo que Merge hace queda expresado mediante la siguiente serie: O_s, $M'O_s$, $M'M'O_s$… Esto puede definirse recursivamente de manera formal como sigue (Mota, 2013, nota 2; 2014, p. 38):

$M^{(0)}'(O_s) = O_s$, Def,
$M^{(n)}'(O_s) = M'M^{(n-1)}'(O_s)$ Def.

Pero el procedimiento empleado por Chomsky y definido por medio de este sistema de ecuaciones recursivas, no solo genera expresiones sin auto-inclusión de constituyentes (como *Juan canta muchas canciones*), sino que también genera estructuras con auto-inclusión como *el ratón que el gato que el perro perseguía mordió corría*, la cual exhibe inclusión central y, además, auto-inclusión, dado que una cláusula de relativo se incluye dentro de otra del mismo tipo. Antes de continuar, conviene advertir que según lo que acabo de exponer, un dominio sin auto-inclusión también puede ser definido por recursión y, por tanto, como ya he mostrado, la

auto-inclusión no justifica la recursión y, por ello, ambos conceptos no deberían identificarse/equiparase.

Dicho esto, la descripción esquemática de cómo genera *Merge* ambas oraciones es como sigue:

Juan canta muchas canciones

M_{Juan}'M_{canta}'$Os_{\{muchas,\ canciones\}}$

el ratón que el gato que el perro perseguía mordió corría

M_{el}'$M_{ratón}$'M_{que}'M_{el}'M_{gato}'M_{que}'$M_{\{\{el,\ perro\},\ persegu\acute{i}a\}}$'$M_{mordió}$'$O_{s\ corría}$

Por otra parte, Chomsky (2007a, p. 6; 2008, p. 139) ha insistido en que el modo de operar (cómo procede, o puede implementarse de manera abstracta) de este procedimiento generativo consiste en aplicar iterativamente la operación *Merge*. Así, la aplicación iterativa queda descrita mediante la siguiente serie, que muestra, en cada paso, las n veces que Merge se aplica de forma sucesiva, lo cual puede generar todo tipo de expresiones (con auto-inclusión o no) del lenguaje natural: $M^{(0)}$, $M^{(1)}$, $M^{(2)}$,…, $M^{(n)}$. Vemos, una vez más, que las estructuras con auto-inclusión no justifican la propiedad de la recursión.

Así, en el caso de la oración *Juan canta muchas canciones* (una oración que no presenta constituyentes dentro de constituyentes del mismo tipo y, por tanto, sin auto-inclusión a tal nivel), la aplicación iterativa es como sigue:

Juan canta muchas canciones

M_{Juan}'M_{canta}'$Os_{\{muchas,\ canciones\}}$

En donde la secuencia completa es:

{A,B} = C; {C,D}= E; {E,F} = G

Que equivale a la regla $M'M'M'(O_s) = M^{(3)}'(Os)$

Esto indica que la operación *Merge* se ha aplicado tres veces sucesivamente (iterativamente) sobre su último resultado generado (Mota, 2013, §1; 2014, p. 27).

Por su parte, en el caso de la oración *el ratón que el gato que el perro perseguía mordió corría* (la cual presenta auto-inclusión) la generación iterativa sería como sigue:

El ratón que el gato que el perro perseguía mordió corría

M_{el}'$M_{ratón}$'M_{que}'M_{el}'M_{gato}'M_{que}'$M_{\{\{el,\ perro\},\ persegu\acute{i}a\}}$'$M_{mordió}$'$O_{s\ corría}$

Cuya secuencia completa es:

{A,B} = C; {C,D} = E; {E,F} = G; {G,H} = I; {I,J} = K; {K,L} = M; {M,N} = P; {P,Q} = R; {R,S} = T

Esto equivale a $M'M'M'M'M'M'M'M'M'(Os) = M^{(9)'}(O_s)$

En este caso, *Merge* se aplica nueve veces de manera sucesiva sobre el último resultado producido, generando iterativamente una expresión jerárquicamente estructurada como la empleada en el ejemplo.

En todos estos ejemplos, es importante tener presente que la recursión es una propiedad de la regla; pues no debemos obviar el hecho de que una definición recursiva como '$M(n)'(O_s) = M'M^{(n-1)}'(O_s)$' es una regla.

En todo este análisis me he ceñido al uso original de 'recursión', como una propiedad constituida por una regla y diferenciándola de la noción de auto-inclusión aplicada a estructuras. Tal y como se desprende de mi análisis, sólo la regla, pero no sus valores o resultados, es recursiva (*Cf.* Zwart, 2011 para otras interpretaciones).

Los resultados o las estructuras poseen sus propiedades independientemente de tales reglas. De hecho, las propias definiciones no dejan duda en este punto. Por repetirlo una vez más, una regla es recursiva si para computar un valor hace uso de valores previamente computados para argumentos menores (ver supra). Por el contrario, una estructura de datos con un componente X dentro de otro X del mismo tipo es lo que se conoce como estructura con auto-inclusión (Arsenijević y Hinzen, 2012), cuyas propiedades son independientes de las reglas recursivas (véase por ejemplo Luuk y Luuk, 2011; Mota, 2013, 2014; Lobina, 2014a, 2014b). A este respecto, parece que se ha establecido una convención según la cual se ha de llamar 'recursión' a la auto-inclusión.[20] Esta convención sobre la auto-inclusión implica (I) el uso de la auto-inclusión para caracterizar las reglas y (II) el uso de la recursión para caracterizar a las estructuras (advierto al lector de que este punto también abarcará el análisis efectuado en la sección siguiente).

En el primer caso, conviene señalar que no es una aseveración adecuada decir que lo que hacen las reglas de rescritura puede caracterizarse mediante la auto-inclusión; como parece indicar Fitch (2010, p. 79), cuando señala que una regla recursiva es una que tiene la propiedad de la auto-inclusión. Veamos varios argumentos que critican esta idea.

Primero, lo que sencillamente hacen es definir/generar una nueva descripción estructural desde una previamente generada (Post, 1921). En este

[20] Ya Soare (1996, 1999, 2009) advirtió que en la Teoría de la Computabilidad se había consolidado una convención que implicaba usar recursión para significar lo mismo que computabilidad (ver supra) y, por tanto, emergió un período de confusión conceptual. Tal Convención de la Recursión (Recursion Convention) implicaba lo siguiente: La 'Recursion Convention' hace referencia a: (1) el uso de términos del formalismo de las funciones recursivas generales para describir resultados (i.e., como 'recursivamente enumerable'), (2) el uso del término 'Tesis de Church' para referirse a un conjunto de diferentes tesis, (3) la denominación de la materia con el lenguaje de la recursión.

sentido, Post no parece especificar en sus obras las reglas con la propiedad de la auto-inclusión, por lo que no es adecuado relacionarlas con dicho término, tal como indico en otro trabajo (Mota, 2014). Así, para generar Σ, ε, εε, εεε, equivalente a Σ, $R(\Sigma)$, $R(R(\Sigma))$, $R(R(R(\Sigma)))$, se puede aplicar una regla como εΣ→εεΣ, lo cual sólo indica que lo que hace tal regla es rescribir las expresiones de lado izquierdo como las del lado derecho; lo que puede formularse como εεΣ = $R'(\varepsilon\Sigma)$ (Mota, 2014, p. 40). Esto último, al igual que las reglas del tipo '$a+b' = (a+b)+1$' sólo muestran que un valor se computa haciendo uso de valores previamente computados y que un valor se substituye/reemplaza por otro, por lo que la noción de auto-inclusión no es adecuada para caracterizar las reglas del cálculo. Esto es igualmente válido para una regla como AB→AABB.

Por su parte, Lobina (2014a, 2014b) señala que Fitch (2010) más que interesarse por reglas recursivas, lo hace por las estructuras con auto-inclusión, dado que entiende que un indicador empírico de la noción de recursión es la interpretación adecuada de una oración con auto-inclusión. Ciertamente, esta cuestión tiene que ver más con una noción puramente semántica y no tanto con una noción sintáctica (Lobina, 2014b, p. 66).[21]

El otro punto de lo que he llamado la *convención sobre la auto-inclusión* tiene que ver con *el uso de 'recursión' para caracterizar a las estructuras* en las que un X contiene o está dentro de otro X del mismo tipo (esto es, en términos generales, una estructura con auto-inclusión, *Cf.* Karlsson, 2010; Arsenijević y Hinzen, 2012; sea X un sintagma, un esquema o una estructura en árbol). La noción original de recursión, surgida en la Lógica Matemática y la Teoría de la Computabilidad hace referencia a una definición por medio de la cual se define un nuevo valor para un nuevo argumento usando un valor previamente definido/computado/generado para un argumento menor. En otros dominios, como en la Ciencia de la Computación, se definen procedimientos y algoritmos recursivamente (lo cual deriva de lo dicho anteriormente) pero también aplican la noción de recursión a una configuración de datos; esto es, a la configuración/organización interna de una expresión como, por ejemplo, '$(a+b)+c$' (ver infra y también Wirth, 1986 para otros ejemplos). En el próximo apartado examinaré este aspecto más detenidamente dentro de la Ciencia de la Computación, una vez analizado el fenómeno en el dominio de la Ciencia Cognitiva.

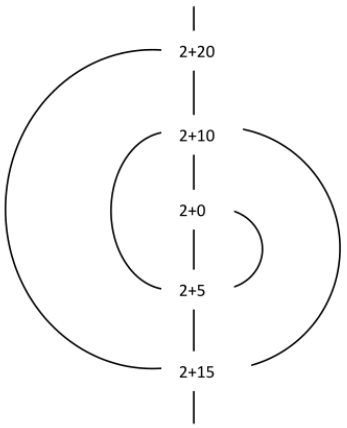
Recursión Vs. Auto-inclusión en la Ciencia de la Computación

[21] Por lo que Fitch (2010, pp. 80-81) parece interesarse es por la manera en la que se asigna un determinado significado a una señal o secuencia particular (y viceversa).

Como he señalado anteriormente, una estructura se considera recursiva si y sólo si un componente X contiene/incluye/está dentro de otro componente X del mismo tipo, siendo una instancia de tal definición las estructuras con auto-inclusión tratadas en el apartado precedente y las que se tratarán en este.

Quisiera advertir que una diferencia entre esta concepción de la recursión y la derivada de una *definición por recursión* (*o definición recursiva)* es que sólo en la primera pero no en la segunda se da una relación de 'estar dentro de', esto es, una estructura recursiva necesariamente tiene que tener un X que contiene o incluye a otro X del mismo tipo.

Creo que esta cuestión la expresó Wittgenstein de una manera meridianamente clara, contribuyendo de manera notable al desarrollo de la lógica matemática y la teoría de la computabilidad (véase, Wittgenstein, 1974, 1975, 1978; Wrigley, 1977; Shanker, 1987; Frascolla, 1994; Marion, 1995, 1998, 2009; Rodych, 1997, 1999a, 1999b, 2003; Oddifreddi, 2001; Mota, 2013, 2014). Como Wittgenstein señaló en las *Observaciones Filosóficas* (1975, §163) con absoluta precisión, una definición recursiva (por ejemplo 'a+0 = a'; 'a+(b+1) = (a+b)+1') es una regla fundamental del sistema que nos indica cómo proceder y, como tal, no se puede aseverar o negar, es decir, no son descripciones y, por ello, carece de valor de verdad. Abundando en esta misma idea, Wittgenstein nos indica en la *Gramática Filosófica* que la definición recursiva "es una regla para la construcción de reglas de sustitución, o también el término general de una serie de definiciones [formas]" (1974, 36, p. 851).[22] De forma muy elocuente, Wittgenstein (1975, §164) expresa gráficamente lo enunciado anteriormente mediante la siguiente figura:



[22] La noción de regla recursiva es un puente entre los primeros trabajos de Wittgenstein y sus posteriores escritos. Así, la forma general de una serie de formas, por ejemplo, '[0, ξ, ξ + 1]', expresa una regla (gramatical) del tipo '(a+ (ξ + 1)) = (a+ ξ)+1' (Mota, 2013, 2014, p. 35).

Esta figura muestra, por ejemplo, el paso (la regla) 'hacia delante' de un término a otro de la serie 2+0, 2+1, 2+2,…,2+5,… pero también muestra el paso (la regla) 'hacia atrás', por ejemplo 2+5 = (2+4)+1.

Por otra parte, una cosa es decir que el término 'estructura' se define inductivamente, lo cual no hace *ipso facto* a las estructuras recursivas, y otra muy diferente decir que una estructura es recursiva atendiendo a la organización interna de sus elementos.[23]

Podemos ver un ejemplo muy claro de esto en el trabajo de Lobina (2014a). Dicho autor, recoge la definición de Roberts (2006) que indica que una clase puede ser recursiva en el sentido de que la definición de esta clase puede hacer referencia a objetos de la misma clase (véase más abajo la definición de 'número natural').[24] Además, Lobina (2014b, p. 59) advierte que Rodgers y Black (2004) definen una estructura recursiva como aquella que está parcialmente compuesta por instancias más simples o pequeñas de la misma estructura de datos. Por otro lado, Wirth (1986, p. 109), nos indica que una estructura de datos es recursiva si un componente contiene a otro del mismo tipo. Así, en el caso de Roberts, la recursión es una propiedad de la definición, no de la clase, mientras que en el caso de Rodgers y Black, lo mismo que en el de Wirth, la recursión es una propiedad de la organización interna de la estructura. Así, se habla en el primer caso de una definición inductiva, mientras que en el segundo se habla de una estructura recursiva; dos cosas muy diferentes que *a priori* no deben confundirse. Veamos unos ejemplos.

Se puede definir inductivamente el término 'número natural' como sigue: (a) 0 es un número natural, (b) el sucesor de un número natural es un número natural. Asimismo, se puede definir el término 'estructura en árbol' de la siguiente manera (véase Wirth, 1986): (a) *O* es una estructura en árbol, llamada vacía, (b) si t_1 y t_2 son estructuras en árbol, entonces las estructuras consistentes en un nodo con dos descendientes $t1$ y t_2 son también estructuras en árbol binarias. En estos casos, la recursión es una propiedad que se constituye en la definición, pero no es una propiedad de los números naturales o las estructuras. Por tanto, la recursión es una propiedad constituida por la regla (Mota, 2013, 2014).

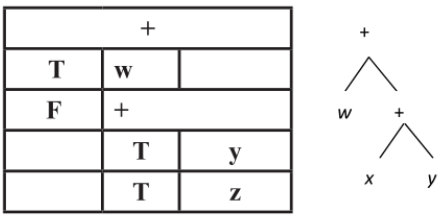
Nótese, por tanto la siguiente diferencia: (I) el término 'estructura' es definido inductivamente y la recursión es una propiedad de la regla, no de la estructura definida y (II) una estructura de datos es recursiva si un

[23] Por 'estructura' se significa, en un sentido general, una organización determinada entre uno o varios elementos.

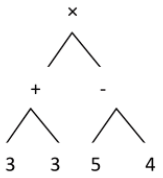
[24] Obsérvese que una serie "recursiva" no es más que una expresión de la regla/definición recursiva. En este caso, la recursión es una propiedad de (o constituida por) la regla.

componente X contiene/incluye o está dentro de otro X del mismo tipo que él. Según esto, la definición recursiva, propia de la lógica matemática y de la teoría de la computabilidad cae fuera de (II) dado que: (a) 'recursión' en (II) se predica de estructuras, algo que no sucede en las mencionadas disciplinas (aunque sí en la Ciencia Cognitiva y de la Ciencia de la Computación) y (b) hace referencia a la configuración/organización interna de una estructura; lo cual no tiene sentido atribuir a una regla que sólo indica cómo proceder en la definición. Es decir, la relación entre los elementos que configuran la regla recursiva y los elementos que configuran la estructura de datos es diferente. Veámoslo con ejemplos.

La ecuación '2+3 = (2+2)+1' es una instancia de una regla recursiva (o una instancia del paso recursivo de una definición recursiva, ver supra), mientras que la organización interna de la expresión '(x+y)+w'



o de la expresión '(3+3)×(5-4)':



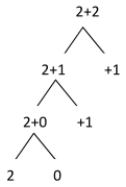
son sendos ejemplos de lo que se entiende en la Ciencia de la Computación como estructuras recursivas de datos: X dentro de X del mismo tipo (algo diferente a una definición recursiva).

Vuelvo a insistir entre la diferencia de una estructura recursiva y un procedimiento recursivo. En éste último caso, y como he señalado arriba, un procedimiento puede aplicarse recursivamente para computar '2+2', invocando un valor previamente computado para un argumento menor, esto es, '(2+1)+1' hasta que alcanza el caso base '(((2+0)+1)+1'. Esta implementación recursiva deriva directamente de la definición recursiva de la función suma (*Cf.* Abelson et al., 1996, entre otros):[25]

[25] Tanto Wirth (1986) como Abelson et al. (1996) indican que aunque una función se defina recursivamente ésta no tiene que implementarse necesariamente de manera recursiva (ver supra).

1. 2+2
2. (2+1)+1
3. (((2+0)+1)+1
4. (2+1)+1
5. 3+1
6. 4

Nótese que éste uso de recursión es diferente al que se predica de la forma de un proceso de cómputo, siendo tal una estructura recursiva de datos (un componente de la estructura en árbol incluye/contiene otro del mismo tipo):

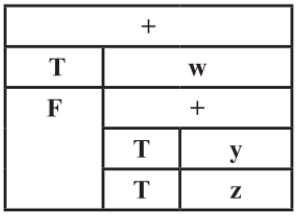


pero son 'recursivos' por razones diferentes; a saber, una regla permite substituir o reemplazar una expresión (2+2) por otra ((2+1)+1) mientras que una estructura tal exhibe la ya mencionada organización interna referente a un X (un componente) dentro de/contiene a otro X (componente) del mismo tipo. De hecho, Wirth (1986, p. 110) indica que una expresión como 2+2 exhibe una estructura recursiva como la siguiente:

+	
T	**y**
T	**z**

Esto es así de manera independiente de la regla recursiva, por lo que una vez más se concluye que la organización interna de una expresión como '2+2' o '(*a*+*b*)+*c*' no justifica la propiedad de la recursión entendida en su sentido original, como propiedad constituida por una regla (Mota, 2013, 2014). Dicho de otro modo, una regla no es recursiva porque una expresión que aparece en la misma exhiba una organización interna como las mostradas aquí.

Por otro lado, aunque en el caso de la regla no esté justificado hablar de auto-inclusión, sí lo está en el caso de las estructuras recursivas, pues no son las funciones las que se caracterizarían mediante esa propiedad, sino el esquema, la estructura que contiene un X dentro de otro X del mismo tipo; esto es, esquemas/componentes dentro de esquemas/componentes del mismo tipo, como en:



Esto es análogo a lo que se puede predicar de los constituyentes o de la configuración de E-N-C definidas por Moro (ver supra), siendo una instancia de un esquema dentro de otro de exactamente del mismo tipo. Si, como vengo mostrando, la recursión en su sentido original no hace referencia a la organización interna de un X dentro de otro X del mismo tipo, propongo reservar la noción de recursión a su sentido original y aplicar en estos otros casos la denominación más apropiada de 'auto-inclusión', estando, así, de acuerdo con Luuk y Luuk (2011, p. 9) en este punto. Nótese bien que esta propuesta es mucho más que una simple cuestión terminológica.

Addendum sobre recursión y auto-referencia

No es infrecuente encontrar en la literatura sobre recursión que tal propiedad es equiparable o se puede identificar con la auto-referencia (Tomalin, 2006, Lobina, 2011, 2014a, 2014b). La auto-referencia se puede definir, siguiendo a Odifreddi (2001), en términos de notación de conjuntos, como '$x \in x$', que puede traducirse como *'x* pertenece a sí mismo'. Un conocido ejemplo de esta formulación general es la Paradoja de Russell sobre la clase que contiene a todas las clases que no pertenecen a sí misma ($R = \{x_1, x_2, x_3, \ldots x_n\}$; pudiendo estar el conjunto vacío). Si por ejemplo la clase x_1 fuera miembro de sí misma (R), entonces no sería miembro de sí misma ($\neg(x \in x)$), pero si no es miembro de sí misma ($\neg(R \in R)$) entonces sería miembro de sí misma ($(R \in R)$): contradicción.

Como he venido señalando, en el caso de las estructuras que exhiben un X dentro de otro X del mismo tipo también cabría hablar de auto-referencia (véase también Lobina, 2011, 2014a, 2014b). Así, una estructura con un X dentro de un X del mismo tipo, esto es, una estructura con auto-inclusión (Arsenijević y Hinzen, 2012), es una estructura que exhibe auto-referencia. Si esto es así, entonces se sigue de la premisa expresada más arriba (esto es, que *x* es recursivo porque *x* es auto-referente) que tal estructura es recursiva. De esto se sigue, por ende, que la auto-inclusión es un tipo de auto-referencia y por tanto un tipo de recursión. Mi intención es hacer una reflexión conceptual entre las nociones de 'recursión' y 'auto-referencia' y ver en qué sentido podemos decir que x es recursivo porque x es auto-referente.

En el caso de las funciones parece que la relación no es exactamente igual que en el caso anterior. Así, no se puede decir que una función *f* contiene otra función f del mismo tipo. Siguiendo a Wittgenstein (1922), en el *Tractatus Logico-Philosophicus* argumentó en respuesta a la Paradoja de Russell que una función no puede contenerse a sí misma; esto es, no puede ser su propio argumento, dado que el signo para función '$F(\ )$' contiene la protofigura de su argumento, y no puede contenerse a sí misma (3.333). Así, en '$F(F(x))$' la función externa (de segundo orden o nivel) e interna (de primer orden), aun usando ambas la misma letra, tienen significados distintos. Es evidente pues que son funciones de diferente tipo. Así, la función interna, nos dice Wittgenstein, tiene la forma $\phi(x)$ y la externa la forma $\psi(\phi(x))$ (Mota, 2013, 2014).

Esta relación de (auto-)inclusión diferencia claramente la recursión que se predica como propiedad constituida por una definición (esto es, por una regla) de la que se predica de la organización interna de una estructura gramatical o de datos (ver supra). Así, en el caso de las estructuras, la recursión se predica de la inclusión de X dentro de X del mismo tipo, sean X sintagmas, esquemas, estructuras en árbol,…, mientras que en el caso de las reglas o las definiciones (por ejemplo de una función o de un proceso) no existe tal relación de inclusión, sino que una expresión se substituye por otra que está constituida por un valor definido con anterioridad para un argumento menor (Mota, 2013, 2014).

Me gustaría terminar este apartado indicando dos cosas. La primera es llamar la atención sobre la extensión de 'recursión' y 'auto-referencia'. Así, si equiparamos 'recursión' y 'auto-referencia', entonces nos encontramos con que expresiones como 'Yo miento', 'Yo soy indecidible' o 'esta expresión tiene cinco palabras' serían recursivas puesto que son auto-referentes. Es cierto que la auto-referencia en estos casos es semántica más que sintáctica pero no se ha especificado en la literatura que tenga que restringirse necesariamente a esto último. Si decimos que todo aquello a lo que subyace la auto-referencia es recursivo entonces la noción de recursión se usará en un sentido muy diferente al original. Esto es, no sólo se predicará de objetos de los que no se predicaba en origen sino que adoptará otros significados distintos (por ejemplo el que se refiere a la auto-referencia semántica).

La segunda cuestión que quería indicar es que aunque se pueda decir que la recursión usada en su sentido original y la recursión entendida como X dentro de X del mismo tipo (esto es, auto-inclusión) presentan algún tipo de auto-referencia, yo reservaría, por lo dicho anteriormente, el término 'recursión' para expresar su sentido original y emplearía el de 'auto-inclusión' en el segundo caso. Esto lo indico por dos razones: (I) no todo lo auto-referente

es recursivo, entendiendo por 'recursivo' su significado primario y original: como acabo de indicar, 'recursión' adquiriría un significado diferente cuando se usara para indicar, por ejemplo, la auto-referencia propia de la Paradoja de Russell; y (II) cuando Soare propuso usar 'recursión' en su sentido original y no como sinónimo de computabilidad (ver supra) lo hizo, entre otras cosas, para no confundir tal sentido original. Así, por ejemplo, la auto-referencia, por lo demás legítima, de una expresión como 'esta expresión tiene cinco palabras' no indica, a la luz de lo expuesto anteriormente, recursión en su sentido original.[26] Por ello, llamo la atención sobre la relación entre 'recursión' y 'auto-referencia', dado que quizá, no todo lo que presenta auto-referencia sea recursivo si queremos preservar el sentido original del término.

Conclusiones

En este trabajo he analizado el uso que del término 'recursión' se hace en los dominios de la Lógica Matemática y la Teoría de la Computabilidad, de la Ciencia Cognitiva y, finalmente, de la Ciencia de la Computación. Se pueden establecer dos usos derivados, por una parte, de su aplicación a la caracterización de un modo de definición, esto es, de una regla (el cual hace referencia a su uso original y primario), y por otra, de su aplicación a la configuración interna de una estructura. De forma similar, he analizado la noción de recursión y su relación con la auto-referencia, indicando que, quizá, no todo lo que presenta auto-referencia sea recursivo, al menos si se entiende 'recursión' en su sentido original y primario. Derivado del análisis, concluyo, de forma similar a como hicieran Luuk y Luuk (2011), que se podría reservar, sin menoscabo conceptual alguno, la noción de recursión a su sentido original y usando, por ejemplo, auto-inclusión, para referirnos a una configuración interna de estructuras con X dentro de X del mismo tipo.

Referencias

Abelson, H. & Sussman, G. J. with Sussman, J. (1996). *Structure and interpretation of computer programs*. Cambridge, MA: MIT Press.

Arsenijevic, B. & Hinzen, W. (2010). Recursion as a human universal and as a primitive. Biolinguistics, 4(2-3), 165-173.

Arsenijević, B. & Hinzen, W. (2012). On the abscence of X-within-X recursion in human grammar, *Linguistic Inquiry*, 43, 423-440.

[26] Para Corballis (2011) "Pienso luego existo" es un ejemplo de recursión, entendida como un X (un pensamiento nos dice él) dentro de sí mismo. Siguiendo mi análisis, este ejemplo exhibe recursión entendida como X dentro de X del mismo tipo (esto es, auto-inclusión) y auto-referencia. Sin embargo, este sentido estructural no es el mismo que el de la definición recursiva. Este es un ejemplo de, en mi opinión, una generalización de la noción de recursión.

Boolos, G. & Jeffrey, R. (1974). *Computability and logic*. Cambridge: Cambridge University Press.
Boolos, G. (1971). The iterative conception of set, *The Journal of Philosophy*, 68, 215-231.
Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control*, 2, 137-167.
Chomsky, N. (1965). *Aspects of theory of syntax*. Cambridge. MA: MIT Press
Chomsky, N. (1995). *The minimalist program*. Cambridge. MA: MIT Press.
Chomsky, N. (2006). *Language and mind*. Cambridge: Cambridge University Press.
Chomsky, N. (2007a). Of minds and language, *Biolinguistics*, 1, 9-27.
Chomsky, N. (2007b). Approaching UG from below. In U. Sauerland & H. M. Gärtner (Eds.), *Interfaces + Recursion = Language?* (pp. 1-30). Berlin: Mouton.
Chomsky, N. (2008). On phases. In R. Freidin, C. Otero, & M. L. Zubizarreta (Eds.), *Foundational issues in linguistic theory* (pp. 133-166). Cambridge. MA: MIT Press.
Chomsky, N. (2010). Some simple evo devo theses: How true might they be for language? In R. Larson, V. Déprez & H. Yamakido (Eds.), *The evolution of human language* (pp. 45-62). Cambridge: Cambridge University Press.
Chomsky, N. (2011). Language and other cognitive systems. What is special about language?. *Language Learning and Development*, 7, 263-278.
Chomsky, N. (2012). Some core contested concepts. *Proceedings of the CUNY 2012*, 1-18.
Church, A. (1932). A set of postulates for the foundation of logic, The Annals of Mathematics, 33, 346-366.
Church, A. (1936). An unsolvable problem of elementary number theory. In M. Davis (Ed.), *The undecidable* (pp. 88-107). New York: Raven Press.
Corballis, M. C. (2007). Recursion, language and starlings, *Cognitive Science, 31*, 69-704.
Corballis, M. C. (2011). *The recursive mind: The origins of human language, thought, and civilization*. Princeton, NJ: Princeton University Press.
Cutland, N. (1980). *Computability: an introduction to recursive function theory*. Cambridge: Cambridge University Press.
Epstein, R. & Carnielli, W. (1989). *Computability: computable functions, logic, and the foundations of mathematics*. Pacific Grove, CA: Wadsworth & Brooks/Cole.
Everett, D. (2005). Cultural constraints on grammar and cognition in Pirahã, *Current Anthropology, 46*(4), 621-646.
Everett, D. (2009). Pirahã culture and grammar: a response to some criticisms, *Language, 85*(2), 405-442.
Fitch, T. (2010). Three meanings of recursion: key distinctions for biolinguistics. In R. Larson, V. Déprez & H. Yamakido (*Eds.*), *The evolution of human language* (pp. 73-90). Cambridge: Cambridge University Press.
Frascolla, P. (1994). *Wittgenstein's philosophy of mathematics*. London: Routledge.

Gödel, K. (1931). On formally undecidable propositions of the Principia Mathematica and related systems. I. In M. Davis (*Ed*.), *The undecidable* (pp. 4-38). New York: Raven Press.
Gödel, K. (1934). On undecidable propositions of formal mathematical systems. In M. Davis (*Ed*.), *The undecidable* (pp. 39-74). New York: Raven Press.
Gödel, K. (1964). Postscriptum to Gödel 1931. In M. Davis (*Ed*.), *The undecidable* (pp. 71-73). New York: Raven Press.
Hauser, M., Chomsky, N. & Fitch, T. (2002). The faculty of language: What is, who has it, and how did it evolve?, *Science, 298*, 1569-1579.
Hrbacek, K. & Jech, T. (1999). *Introduction to Set Theory*, New York: Marcel Dekker, Inc.
Jackendoff, R. & Pinker, S. (2005). The nature of the language faculty and its implications for evolution of language (reply to Fitch, Hauser, and Chomsky), *Cognition, 97*, 211-225.
Jackendoff, R. (2011). What is the human language faculty? Two views, *Language, 87*, 586-624.
Karlsson, F. (2010). Syntactic recursion and iteration. In H. van der Hulst (*Ed*.), *Recursion and human language* (pp. 43-67).
Kinsella, A. (2010). Was recursion the key step in the evolution of the human language faculty? In H. van der Hulst (Ed.), *Recursion and human language* (pp. 179-191).
Kleene, S. C. (1938). On notation for ordinal numbers, *The Journal of Symbolic Logic, 3*, 150-5.
Kleene, S. C. (1943). Recursive predicates and quantifiers, *Transactions of the American Mathematical Society, 53*, 41-73.
Kleene, S. C. (1952). *Introduction to metamathematics*. Amsterdam: North-Holland Publishing.
Lobina, D. J. (2011). Recursion and the competence/performance distinction in AGL tasks, *Language and Cognitive Processes*, 26, 1563-1586.
Lobina, D. J. (2014a). Probing recursion. *Cognitive Processing*, DOI 10.1007/s10339-014-0619-z
Lobina, D. J. (2014b). What linguists are talking about when talking about..., *Language Sciences, 45*, 56-70.
Luuk, E. & Luuk, H. (2011). The redundancy of recursion and infinity for natural language, *Cognitive Processing, 12*, 1-11.
Marion, M. (1995). Wittgenstein and finitism, *Synthese, 105*, 141-176.
Marion, M. (1998). *Wittgenstein, finitism, and the foundations of mathematics*. Oxford: Oxford University Press.
Marion, M. (2009). Radical anti-realism, *Wittgenstein and the length of proofs, Synthese, 171*, 419-432.
Moro, A. (2008). *The boundaries of Babel*. Cambridge, MA: MIT Press.
Mota, S. (2013). La propiedad de la recursión en el "Tractatus Logico-Philosophicus" de Wittgenstein y su relación con la Teoría de la Computabilidad y la Lógica

Matemática, 17, *Observaciones Filosóficas*. http://www.obervacionesfilosoficas.net/lapropiedaddelarecursion.htm
Mota, S. (2014). La historia y la gramática de la recursión: una precisión desde la obra de Wittgenstein, *Pensamiento y Cultura, 17*, 20-48.
Odifreddi, P. (2001). Recursive functions: an archeological look. In C.S. Claude, M.J. Dinneen & S. Sburlan (*Eds*.), *Combinatorics, Computability and Logic* (pp. 13-31). London: Springer-Verlag.
Pinker, S. & Jackendoff, R. (2005). The faculty of language: What's special about it?, *Cognition, 95*, 201-236.
Post, E. (1921). Introduction to a general theory of elementary propositions, *American Journal of Mathematics, 43*, 163-185.
Post, E. (1943). Formal reductions of the general combinatorial decision problem, *American Journal of Mathematics, 65*, 197-215.
Post, E. (1944). Recursively enumerable sets of positive integers and their decision problems. In M. Davis (*Ed*.), *The undecidable* (pp. 305-337). New York: Raven Press.
Roberts, E. (2006). *Thinking Recursively with Java*. Hoboken, NJ: John Wiley and Sons, Inc.
Roddych, V. (1999a). Wittgenstein on irrationals and algorithmic decidability, *Synthese, 118*, 279-304.
Rodgers, P., Black, P.E. (2004). Recursive data structure. In: Pieterse, V., Black, P.E. (Eds.), Dictionary of Algorithms and Data Structures. Online at: http://www.nist.gov/dads/HTML/recursivstrc.html.
Rodych, V. (1997). Wittgenstein on mathematical meaningfulness, decidability, and application, *Notre Dame Journal of Formal Logic, 38*, 195-225.
Rodych, V. (1999b). Wittgenstein's inversion of Gödel's Theorem, *Erkenntnis, 51*, 173, 206.
Rodych, V. (2002). Wittgenstein on Gödel: the newly published remarks, *Erkenntnis, 56*, 379-397.
Rodych, V. (2003). Misunderstanding Gödel: new arguments about Wittgenstein and new remarks by Wittgenstein, *Dialectica, 57*, 279-313.
Shanker, S.G. (1987). Wittgenstein versus Turing on nature of Church's thesis, *Notre Dame Journal of Formal Logic, 28*, 615-649.
Sieg, W. (1997). Step by recursive step: Church´s analysis of effective calculability, *The Bulletin of Symbolic Logic, 3*, 154-180.
Skolem, T. (1923). The foundations of elementary arithmetic established by means of the recursive mode of thought, without the use of apparent variables ranging over infinite domains. In J. Van Heijenoort (*Ed*.), *From Frege to Gödel. A source book in mathematical logic, 1879-1931* (pp. 302-333). Cambridge, MA: Harvard University Press.
Soare, R. (1996). Computability and recursion, *The Bulletin of Symbolic Logic, 2*, 284-321.
Soare, R. (2009). Turing oracles machines, online computing, and three displacements in computability theory, *Annals of Pure and Applied Logic*, 160, 368-399.

Tomalin, M. (2006). *Linguistics and the Formal Sciences*. Cambridge: Cambridge University Press.
Tomalin, M. (2007). Reconsidering recursion in syntactic theory, *Lingua, 117*, 1784-1800.
Tomalin, M. (2011). Syntactic structures and recursive devices: A legacy of imprecision, Journal of Logic, *Language and Information, 20*, 297-315.
Turing, A. (1937). On computable numbers, with an application to the Entscheidungsproblem. In M. Davis (*Ed.*), *The undecidable* (pp. 116-151). New York: Raven Press.
Wirth, N. (1986). *Algorithms and data structures*. Hemel Hempstead, UK: Prentice Hall. © N. Wirth 1985 (Oberon version: August 2004).
Wittgenstein, L. (1922). *Tractatus logico-philosophicus*. London: Routledge.
Wittgenstein, L. (1974). *Philosophical grammar*. Traducción de Luis F. Segura, ed. 1992. México, D.F.: UNAM.
Wittgenstein, L. (1975). *Philosophical remarks*. Traducción de Alejandro Tomasini Bassols, ed. 1997. México, D.F.: UNAM.
Wittgenstein, L. (1978). *Remarks on the foundations of mathematics*. Oxford: Blackwell.
Wrigley, M. (1977). Wittgenstein's philosophy of mathematics, *Philosophical Quarterly, 27*, 50-59.
Zwart, J.W. (2011). Recursion in language: A layered-derivation approach, *Biolinguistics, 5*, 43-56.

ARTÍCULO 2

Mota, S. (2015). ¿Por qué se usa 'recursión' cuando se quiere significar 'auto-inclusión'?: clarificaciones conceptuales sobre la recursión en el programa chomskiano, Revista de Lingüística Teórica y Aplicada, 53, 171-191. (SJR: Q2 *lingüística y lenguaje*; CARHUS+ 2014 Clase [A])

RLA. Revista de Lingüística Teórica y Aplicada
Concepción (Chile), 53 (2), II Sem. 2015, pp. 171-191.

CL ISSN 0033 - 698X

¿POR QUÉ SE USA 'RECURSIÓN' CUANDO SE QUIERE SIGNIFICAR 'AUTO-INCLUSIÓN'?: CLARIFICACIONES CONCEPTUALES SOBRE LA RECURSIÓN EN EL PROGRAMA CHOMSKIANO[1]

WHY 'RECURSION' IS USED WHEN 'SELF-EMBEDDING' IS MEANT?: CONCEPTUAL CLARIFICATIONS ON RECURSION IN THE CHOMSKYAN PROGRAM

SERGIO MOTA
Departamento de Psicología Básica
Universidad Autónoma de Madrid, España
sergio.mota.v@gmail.com

RESUMEN

Este trabajo pretende presentar de forma clara y concisa la (doble) disociación conceptual entre 'recursión' y 'auto-inclusión' tan frecuentemente confundidos en la literatura especializada. Se mostrará que la recursión no implica auto-inclusión y que la auto-inclusión no implica tampoco recursión. El ámbito de discusión será el programa chomskiano, dado que ha sido uno de los ámbitos donde la recursión ha sido peor interpretada, fundamentalmente por sus críticos.

Palabras clave: Recursión, auto-inclusión, regla, estructura, Chomsky.

ABSTRACT

This work intends to show in a clear and concise way that there is a double conceptual dissociation between 'recursion' and 'self-embedding', two terms which are frequently conflated in the specialized literature. It will show that recursion does not imply self-embedding and that self-embedding does not imply recursion either. The discussion

[1] Mi gratitud al profesor Noam Chomsky por su dedicación e interés por los análisis presentados, especialmente en el último punto de este trabajo. Asimismo, quiero expresar mi agradecimiento a los profesores José Manuel Igoa y Luis Eguren, quienes han leído detenidamente el manuscrito y han hecho inteligentes comentarios que han contribuido a mejorar la calidad de este trabajo. Este trabajo se enmarca parcialmente bajo el proyecto PSI2012-37623, financiado por el Ministerio de Economía y Competitividad del Gobierno de España, dada mi estrecha colaboración con el profesor José Manuel Igoa.

will be focused on the Chomskyan program, since this has been one particular domain where recursion has been worse understood, especially by Chomsky's critics.

Keywords: Recursion, self-embedding, rule, structure, Chomsky.

Recibido: 04.12.2014. Aceptado: 19.10.2015.

1. INTRODUCCIÓN

El presente artículo, lejos de querer presentar una detallada exégesis de la obra de Noam Chomsky, se limitará a hacer una breve exposición de la importancia que la noción de recursión ha tenido y tiene en su obra. Pero antes de presentar estos breves apuntes, conviene advertir que la noción de recursión ha recibido diversas interpretaciones en la bibliografía especializada, incluso en aquellos trabajos que tratan de precisar su significado. En esta introducción se expondrá el significado original del concepto de recursión en la Lógica Matemática y la Teoría de la Computabilidad, y se hará un breve repaso del desarrollo de diferentes clases de funciones recursivas, avances que están, como el propio Chomsky reconoce en sus obras, en la base de la formulación del procedimiento generativo/computacional que subyace a la facultad del lenguaje. Esta presentación está motivada porque autores reconocidos en el ámbito de la Lingüística que han trabajado sobre este tópico, como es el caso de Tomalin (2007, 2011), tienden a confundir la noción de recursión con las diferentes clases de funciones formuladas. Por tanto, se entiende que lo que varía son las clases, pero no la noción de recursión como definición por inducción (ver *infra*)[2]. En el caso particular de este autor, esta confusión le ha llevado a formular hasta nueve significados diferentes del concepto de recursión, de los cuales seis tienen que ver con los avances en el campo de la Teoría de la Computabilidad (2011, p. 307). Sin embargo, en dicho campo, el significado original y primario de recursión se aplica exactamente en el mismo sentido a diferentes clases de funciones (cf. Kleene, 1943).

[2] Como Kleene (1943) muestra, entre otros, todas las clases de funciones recursivas –i.e., las primitivas, las generales y las parciales– hacen uso del principio de recursión. Así, el esquema V, que hace uso de tal principio, es empleado para definir las distintas clases de funciones. Además, se emplean los esquemas I-IV –que aluden, respectivamente, a la función sucesor, a las funciones constantes, a las funciones identidad como funciones iniciales, y al esquema de substitución– y el esquema VI en el caso de las generales y las parciales –que alude al esquema de *minimalización*–. La clase de las funciones recursivas generales se cierra bajo I-VI, teniendo que cumplir la función definida mediante el esquema VI la exigencia de que tiene que tener al menos una solución. Al relajar esta exigencia tenemos las funciones recursivas parciales, también definidas mediante I-VI, pero, en este caso, una función definida mediante VI no tiene que tener necesariamente una solución. Un apunte interesante es que el esquema V presenta diversas *versiones*: una versión en la que la inducción se aplica sobre una variable, otra en la que se emplean parámetros, otra en la que se aplica la inducción sobre, al menos, dos variables simultáneamente.

Como es bien sabido, la noción de recursión encuentra su origen en el siglo XIX, cuando Richard Dedekind y Giuseppe Peano emplearon tal noción en su sentido primario para hacer referencia a una definición de funciones y predicados que, como indica Soare (1996, 1999, 2009), se puede expresar del siguiente modo: una función está técnicamente definida por recursión si y sólo si ésta se define para un argumento X haciendo uso de sus propios valores previamente computados (para argumentos menores que X); pudiendo emplearse también funciones previamente definidas (véase también Gödel, 1931; Kleene, 1952, 2002; Cutland, 1980).

En este sentido, en un texto clásico de este campo, Epstein y Carnielli (1989) indican que, en su forma más simple, la definición recursiva de una función f sería como sigue (siendo g una función previamente definida): $f(0) = m$; $f(n+1) = g(f(n))$ (cf. Kleene, 1952, 2002; Boolos y Jeffrey, 1974). Otro ejemplo clásico, en donde se hace uso del principio de recursión con parámetros, es el de la función suma:

$a+0 = a / a+1 = a'$ Def. [caso base]
$a+(b+1) = (a+b)+1$ Def. [paso recursivo]

Una función como ésta es una función que pertenece a la clase de las funciones recursivas *primitivas*, definidas y empleadas por Skolem (1923) y Gödel (1931, 1934). Dado que éstas no agotan todas las clases de funciones totales que pueden definirse por recursión, Gödel (1934) amplió la clase de las funciones recursivas primitivas formulando la clase de las funciones recursivas denominadas *generales*, que incluyen a las primitivas. Una diferencia fundamental entre la clase de las funciones recursivas generales y la clase de las funciones recursivas primitivas es que en las primeras la inducción se aplica, al menos, sobre dos variables simultáneamente. A continuación, se presenta una función recursiva general en donde las funciones ψ y χ están previamente definidas y la función φ es la que se define por recursión (véase Gödel, 1934, p. 69).

$\varphi(0,y) = \psi(y)$,
$\varphi(x+1, 0) = \chi(x)$,
$\varphi(x+1, y+1) = \varphi(x, \varphi(x+1, y))$.

Seguidamente, se presenta una clase de funciones recursivas que Tomalin no menciona, pero que tienen una enorme relevancia en la Teoría de la Computabilidad, a saber, las funciones recursivas *parciales*.

Esta clase de funciones, definida por Kleene (1938, 1943, 1952) –quien, además, identificó correctamente la clase de funciones computables (Kleene, 1952, pp. 323-348)– debe su nombre al hecho de que una función no necesita estar de-

finida para todas las *n*-tuplas de números naturales que toma como argumentos[3]. Por esta razón, las funciones recursivas parciales incluyen a las primitivas y a las generales, identificándolas como aquellas funciones parciales definidas para *todas* la *n*-tuplas de números naturales tomados como argumentos; esto es, como *funciones recursivas totales*[4]. Conviene subrayar que esta clase de funciones recursivas está más relacionada con los posteriores trabajos en la teoría de la gramática de lo que a simple vista pudiera parecer, dado que como recientemente ha traído Chomsky (2012) a colación, desde un punto de vista conceptual se asume que la recursión es necesariamente infinita, pero las funciones recursivas parciales ponen de manifiesto que una función recursiva puede dar un único valor o incluso ninguno, en caso de que no esté definida para un argumento dado.

Para terminar este breve recorrido por la Teoría de la Computabilidad y la Lógica Matemática, es aconsejable aludir a Wittgenstein, cuyas reflexiones acerca de la noción de regla gramatical se hallan –a juicio del autor del presente trabajo–, en la base misma del concepto de recursión (Mota, 2015)[5]. Así, la definición recur-

[3] Un ejemplo de función parcial es: $f(x) = (x\text{-}1)/2$. Esta función sólo está definida para los números naturales impares tomados como argumentos (cf. Boolos y Jeffrey, 1974; Mota, 2015 –corrigiendo en este ejemplo una errata que aparece allí–). Para una definición más técnica véase Kleene (1938, 1943, 1952).

[4] En Mota (2013, 2014, 2015) se define explícitamente la *Tesis de Kleene*, la cual expone que "una función es computable si y sólo si es una función recursiva parcial (o está definida por el formalismo de Kleene)" (2014, p. 24, nota 3), y la *Tesis de Turing-Kleene* que reza así: "*Una función es computable si y sólo si es una función recursiva parcial (o está definida por el formalismo de Kleene) o, de forma equivalente, si es computable/calculable por una máquina de Turing*" (*Ibid.*). Esta última bien podría substituir a la *Tesis de Church-Turing*, dado que, como se señala, fue Kleene, y no Church (véase Soare, 2009), quien identificó correctamente la clase de las funciones computables.

[5] En Mota (2015) se realiza un tratamiento algo más extenso en relación con la noción de recursión: "Así, desde la lógica matemática, la matemática y la ciencia de la computación, se puede distinguir entre las definiciones inductivas y co-inductivas (que proporcionan los principios duales a los de las primeras). Las primeras proporcionan, o están en la base de, dos principios diferentes que, *a priori*, no hay que confundir, los principios de inducción (que permiten probar ciertas propiedades internas de los objetos tratados, por ejemplo de los números naturales o de las fórmulas proposicionales), y de recursión (que, como se verá, garantizan la buena definición de las funciones recursivas), las segundas proporcionan los principios de co-inducción (que también permiten probar ciertas propiedades internas de los objetos tratados) y de co-recursión (que garantizan la buena definición de funciones a través de procesos). Desde un punto de vista formal, estas definiciones se pueden contextualizar de varias maneras: en la Teoría de Conjuntos, en la Teoría de categorías, entre otras. Sin embargo, no es el objetivo de este trabajo entrar en cada una de estas aplicaciones" (p. 156). En ese trabajo se eligió la teoría axiomática de conjuntos de Zermelo-Fraenkel (ZF) con axioma de elección. Por otro lado, conviene no confundir la definición inductiva, la cual sirve para definir términos como 'número natural', 'objeto', etc., con una definición por recursión (o 'recursiva'). Sólo la segunda, y no la primera, es la empleada en la Lógica Matemática y la Teoría de la Computabilidad para definir funciones y/o predicados (Kleene, 1952, p. 217). Dado que aquí se proporcionan varios ejemplos de definiciones recursivas, también es apropiado presentar un ejemplo de definición inductiva. Así, en el marco de ZF con axioma de elección, una definición inductiva de conjunto inductivo es como sigue: un conjunto I se dice inductivo si (a) el conjunto vacío '∅' pertenece a I y (b) siempre que un conjunto X pertenece a I, entonces también el sucesor S(X), obtenido de la unión de X con el conjunto que tiene como único elemento a X, esto es, $S(X) = X \cup \{X\}$. Así, pueden definirse cómodamente el conjunto de los números naturales. Es importante apuntar que en esta definición,

siva, expresada como 'a+0 = a; a+(b+1) = (a+b)+1', es, como Wittgenstein (1975, §163) señala en las *Observaciones Filosóficas*, una regla fundamental del sistema que indica cómo proceder y, como tal, no se puede aseverar o negar, es decir, no son descripciones y, por ello, carece de valor de verdad. Abundando en esta misma idea, Wittgenstein (1974, 36, p. 851) indica en la *Gramática Filosófica* que la definición recursiva "es una regla para la construcción de reglas de sustitución, o también el término general de una serie de definiciones [formas]".

Si bien es verdad que Chomsky no ha reconocido explícitamente su coincidencia con las ideas de Wittgenstein acerca del concepto de regla gramatical (véase por ejemplo Mota, 2014, 2015, para una discusión más amplia sobre este asunto), no es menos cierto que nunca ha sido ajeno al desarrollo de la Teoría de la Computabilidad y la Lógica Matemática y de ello dejó constancia en sus primeros escritos. Así, en *The logical structure of linguistic theory* (1955/1975), Chomsky indica que la Lógica Matemática, en particular la teoría de las funciones recursivas y la metamatemática, fue siendo cada vez más accesible a diferentes disciplinas y los desarrollos en estas áreas proporcionaban herramientas apropiadas para un estudio más preciso del lenguaje natural (p. 39).

Chomsky insiste en varios de sus trabajos, tanto de la primera época como de tiempos recientes, en la importancia de la Teoría de la Computabilidad y la Lógica Matemática. Así, en otro de sus primeros trabajos indica que, al menos desde un punto de vista conceptual, la teoría de la gramática puede ser vista como el estudio de una clase especial de funciones recursivas (1959, p. 149). Una función recursiva no es sino el procedimiento generativo/computacional que subyace a la facultad del lenguaje en la concepción de Chomsky. Mucho más recientemente (Chomsky, 2007a), indica que uno de los principales factores que han impulsado el desarrollo de la Biolingüística ha sido precisamente el trabajo realizado dentro de la Teoría de la Computabilidad, que ha permitido estudiar más seriamente los mecanismos formales de la gramática generativa. En esta misma línea, ha señalado recientemente (Chomsky, 2012) que el estudio del lenguaje-I, entendido como un sistema dotado de infinitud discreta, cae dentro de la Teoría de la Computabilidad, siendo, de este modo, contemplado como un procedimiento computacional que genera recursivamente una cantidad potencialmente infinita de expresiones jerárquicamente estructuradas.

En el siguiente apartado se mostrará cómo la noción de recursión también se ha venido empleando para referirse a las *estructuras* lingüísticas que presentan una determinada organización interna caracterizada por la 'auto-inclusión' (*self-embedding*). No obstante, se argumentará que la noción de recursión aplicada a

un conjunto *no* puede contenerse a sí mismo, con lo que la posibilidad de la auto-inclusión, que caracteriza la auto-referencia tan típica de la paradoja de Russell, queda neutralizada. Ello también es señalado por Boolos (1971, p. 222), cuando indica que ningún conjunto pertenece a sí mismo, puesto que cada conjunto formado en cada fase, contiene sólo a conjuntos formados en fases previas.

las estructuras no se ajusta al significado original del término. Posteriormente, en el punto 3, se discutirá la noción de recursión en el actual Programa Minimista (Chomsky, 1995a), a objeto de dejar claro que la recursión es sólo una propiedad de una regla, de una definición, si se entiende en su sentido original, y que la auto-inclusión no justifica la recursión. En el punto 4 se presentan algunas consecuencias de esta propuesta. Finalmente, se presentarán brevemente las conclusiones finales.

2. LOS SISTEMAS DE PRODUCCIÓN: RECURSIÓN VERSUS AUTO-INCLUSIÓN

En los años 60, Chomsky entendía por 'gramática generativa' un conjunto de reglas que generan cadenas de símbolos (esto es, las sentencias de un lenguaje) a las que se puede dar una descripción estructural (Chomsky, 1965, p. 10). Asimismo, Chomsky y Miller (1963, p. 284) establecen que por 'gramática' entienden un conjunto de reglas que recursivamente especifican las oraciones de un lenguaje. Una definición recursiva formal de tal caracterización es, tal y como ha sido definida en Mota (2014, p. 39), como sigue:

$R^{(0)'}(s) = s$ Def.,
$R^{(n)'}(s) = R'R^{(n-1)'}(s)$ Def.

R está por un conjunto de reglas, *s* por una estructura generada/especificada por esas reglas y *R*(*s*) por una nueva estructura generada/especificada por medio de la aplicación de *R*( ) sobre una estructura previamente generada/especificada *s*.

Esta definición recursiva guarda un gran parecido con los sistemas de producción de Post (1921, 1943, 1944), en los que Chomsky reconoce abiertamente que se apoya (véase 1965). Muy brevemente, el formalismo de Post consiste en un sistema capaz de generar todas las proposiciones de la lógica de enunciados. Así, sea *g* un enunciado de la lógica proposicional y *P* una variable operacional aplicada a dicho enunciado, el sistema produce un enunciado *g'* que sustituye a *g*. Esto se expresa en su sistema de producción normal como sigue: $gP \rightarrow Pg'$ (Post, 1943, p. 199). Su tesis, conocida como la Tesis de Post, establece que un conjunto no vacío (como el de las proposiciones de la lógica de enunciados) es efectivamente numerable si y sólo si es derivado de un sistema de producción (normal) [derivado de su sistema canónico (normal)] (Davis, 1982; Soare, 2009)[6]. Una definición recursiva de un sistema tal puede verse en Mota (2014, p. 31):

[6] Post (1944) también se centró en los conjuntos recursivamente numerables, distinguiendo entre éstos y los conjuntos recursivos. Un conjunto *S* es recursivo si se puede establecer un método efectivo –un procedimiento mecánico finito– para determinar si, por ejemplo, un número entero positivo *n* pertenece o no al conjunto *S*. Un conjunto *S* es recesivamente numerable si existe un procedimiento mecánico efectivo para numerar efectivamente los elementos de *S*.

$P^{(0)}{}'(g) = g$ Def.,
$P^{(n)}{}'(g) = P'P^{(n-1)}{}'(g)$ Def.

Sin embargo, dentro de la Ciencia Cognitiva del Lenguaje, no sólo se predica recursión de un sistema como el que se presentó más arriba, sino de reglas específicas. Así, siguiendo a Chomsky y Miller (1963), un sistema como Σ→εΣ; Σ→ε no expresa sino reglas de un cálculo que permite generar Σ, ε, εε, εεε, lo que equivale a Σ, $R(\Sigma)$, $R(R(\Sigma))$, $R(R(R(\Sigma)))$. Así, para generar εεε, el sistema aplica εΣ→εεΣ (lo cual quiere decir que las expresiones de lado izquierdo se rescriben como las del lado derecho), lo que puede formularse como εεΣ = $R(\varepsilon\Sigma)$ (Mota, 2014). Esto último son reglas del tipo '$a+b'$ = $(a+b)+1$' (Ibíd.). Luuk y Luuk (2011) caracterizan de manera un tanto general las reglas recursivas como aquellas en las que todo aquello que aparece en el lado izquierdo de la flecha reaparece en el lado derecho. Sea como fuere, bajo estas premisas, se entiende que una regla como AB→AABB es una regla recursiva.

Hasta aquí, la recursión se ha mostrado como una propiedad de una definición, referida específicamente a sistemas de reglas o de cómputo en su totalidad. Sin embargo, desde hace unos años, el término 'recursión' se comenzó a emplear para significar una propiedad de la organización interna de las estructuras lingüísticas. Así, una estructura con *auto-inclusión* se define como aquella en la cual un constituyente (o una estructura A) contiene otro constituyente (o estructura B) del mismo tipo (Pinker y Jackendoff, 2005; Moro, 2008; Karlsson, 2010; Kinsella, 2010). Ejemplos generales de estructuras con auto-inclusión (las llamadas estructuras recursivas) son aquellas que exhiben sintagmas dentro de sintagmas del mismo tipo (por ejemplo, un sintagma nominal incluido en otro cf. Karlsson, 2010, p. 51) o también oraciones de relativo dentro de oraciones de relativo (Ibíd.)[7]. Así, Jackendoff (2011, p. 592) afirma que tanto [A[AB]B] como [[[AB]C]D] son estructuras recursivas, entendiendo siempre que la estructura interna y la externa son del mismo tipo, pues éste es el criterio definitorio del término (cf. Luuk y Luuk, 2011). Sin embargo, recursión y auto-inclusión hacen referencia a propiedades muy diferentes y la recursión no está justificada por la auto-inclusión. A continuación se verá por qué.

Como se acaba de señalar, una regla como AB→AABB es una regla recursiva. Tal regla genera cadenas o secuencias de símbolos como AABB, AAABBB,…, (Luuk y Luuk, 2011, p. 5). Por otro lado, diversos autores (Corballis, 2007; Luuk y Luuk, 2011) señalan que una secuencia como AABB no presenta necesariamente auto-inclusión, dado que puede describirse alternativamente como una

[7] Un ejemplo del primer tipo es "*[la casa de [la sierra]$_{SN}$]$_{SN}$ es de piedra*", en la que un sintagma nominal (SN) incluye a otro SN. Un ejemplo del segundo tipo, y que se empleará más abajo, es "*el ratón [que el gato [que el perro perseguía]$_{OR}$ mordió]$_{OR}$ corría*", en donde una oración de relativo (OR) incluye a otra OR.

sucesión de unidades de un tipo (A) seguida del mismo número de unidades de otro tipo (B) (así, [AA][BB]), sin que haya dependencias entre unidades no adyacentes dispuestas jerárquicamente (como en [A[AB]B]). Además, estos autores señalan que una secuencia como AABB, aun cuando presente una estructura con auto-inclusión, no tiene por qué haberse generado necesariamente mediante reglas recursivas. Y a la inversa, aunque la regla AB→AABB sí sea una regla recursiva, la secuencia AAABBB, resultante de su aplicación, no puede juzgarse como recursiva o con auto-inclusión (Ibíd.). Esto compromete, por ejemplo, el gran salto inferencial llevado a cabo por Jackendoff (2011, p. 592) y que queda expresado en los siguientes términos: si se puede establecer el dominio de las estructuras con auto-inclusión (lo que Jackendoff llama *recursión estructural*), entonces se podrá concluir que las reglas que definen este dominio son formalmente recursivas. Sin embargo, tanto [A[AB]B] como [AA][BB] se pueden generar mediante una regla del tipo AB→AABB, de tal manera que si se establece un dominio con estructuras sin auto-inclusión, se podrá concluir que las reglas que definen dicho dominio son recursivas. De ello se desprende que la auto-inclusión no justifica la recursión, y, por tanto, se puede establecer un dominio de recursión formal independientemente de que las estructuras exhiban o no auto-inclusión. En esta misma línea, Fitch (2010, p. 87) afirma que secuencias del tipo AB, ABAB,…, esto es, cadenas de símbolos de las que comúnmente se considera que no presentan inclusión, sino que consisten en secuencias repetidas de AB, son generadas por un sistema como S→AB; S→AB+S, en donde S→AB+S es una regla recursiva; y lo mismo podría decirse del sistema que se presentó más arriba basado en el sistema Σ→εΣ; Σ→ε, el cual genera secuencias o cadenas de símbolos como ε, εε, εεε, de las cuales no se puede decir que presenten auto-inclusión.

De todo lo anterior cabe concluir, por tanto, que la recursión es una propiedad de las reglas y no de las estructuras, en contraste con la extendida opinión de que la recursión es una propiedad de las estructuras. Por consiguiente, cuando se predica recursión de las estructuras, sólo se quiere decir auto-inclusión, esto es, la existencia de constituyentes dentro de constituyentes del mismo tipo.

En relación con este último punto, conviene diferenciar también entre inclusión central (*center-embedding*) y auto-inclusión (*self-embedding*). Así, una estructura exhibe inclusión central cuando un constituyente A está totalmente incluido en B, con elementos no nulos a izquierda y derecha (cf. Chomsky, 1965, p. 12). A este respecto, hay que señalar que para algunos autores (v.g. Karlsson, 2010, p. 51) la inclusión central es la recursión por excelencia. Sin embargo, esto tampoco es correcto, dado que según la concepción "estructural" de la recursión, sólo cabe atribuir esta propiedad a las estructuras que incluyan constituyentes del mismo tipo (véase, por ejemplo, Arsenijević y Hinzen, 2012 o Vicari y Adenzato, 2014, p. 173; trabajos en los que la auto-inclusión se considera una propiedad central de la recursión). Así, una estructura [A[AB]B] no indicaría, para este último grupo

de autores, una estructura recursiva *necesariamente*.

Un análisis similar a éste se puede efectuar desde la Teoría de la Computabilidad y la Lógica Matemática. Así, una ecuación como '$\varphi(a,x) = \psi(\varphi(a,x\text{-}1))$' es un paso recursivo formal en el que no hay rastro de auto-inclusión. Esta ecuación muestra que la expresión del lado izquierdo ($\varphi(a,x)$) se puede *reemplazar/substituir/intercambiar* (pero no auto-incluir) por la del lado derecho ($\psi(\varphi(a,x\text{-}1))$), en donde ψ y φ son dos funciones diferentes. En este caso, la recursión se muestra en el hecho de que para definir un nuevo valor (de φ) se hace uso de un valor previamente computado por esa función para un argumento menor, es decir, se hace uso de la definición por recursión, pero ninguna de las expresiones por separado es recursiva, sino, más bien, la regla. De ahí que una función no pueda contenerse a sí misma, esto es, que no pueda ser ella misma su propio argumento (Mota, 2015).

En el siguiente apartado se va a afianzar esta diferencia entre recursión y auto-inclusión desde la perspectiva del actual Programa Minimista, en el que los sistemas de producción han sido reemplazados por un procedimiento generativo/computacional basado en una única operación: *Merge* (traducida como "ensamble"; véase Eguren y Fernández, 2004).

3. LA RECURSIÓN EN EL ACTUAL PROGRAMA MINIMISTA: UNA PROPIEDAD DEL PROCEDIMIENTO MECÁNICO FINITO

Como ha señalado Chomsky en varios textos (véase por ejemplo, Chomsky, 1995a, 2007b), los sistemas de producción basados en reglas de rescritura (ver *supra*) empleados en los años 50 y 60 para explicar la generación de expresiones del lenguaje natural, se fueron eliminando de la teoría hasta que en los años 80, bajo el modelo de *Principios y Parámetros*, la gramática generativa se centró más en los principios y condiciones de buena formación de las expresiones estructuradas (véase, por ejemplo Chomsky, 1980), que en los procedimientos que dan cuenta de su generación.

Años después, y dentro del *Programa Minimista*, Chomsky (1995a) expurga conceptualmente sus propuestas anteriores y establece que un lenguaje-I es un procedimiento generativo que genera recursivamente una ordenación (o serie) infinita de expresiones jerárquicamente estructuradas mediante la operación denominada *Merge*. Así, lo que hace un procedimiento mecánico finito de este tipo (también llamado sistema computacional o procedimiento generativo) es construir recursivamente objetos sintácticos (Chomsky, 1995a, p. 226). Como es bien sabido, las expresiones u objetos sintácticos generados por *Merge* son interpretados por dos sistemas, a saber: el sensorio-motor y el conceptual-intencional (véase para más detalles Hauser, Chomsky y Fitch, 2002; Chomsky, 2006, 2007b, 2008, 2010, 2011).

Lo que interesa aquí es analizar qué hace *Merge*, que como ya se señaló, es generar recursivamente objetos sintácticos. Así, se puede expresar la forma general de *Merge* de la siguiente manera: [N, O_s, $M(O_s)$] (Mota, 2013, 2014, 2015). N hace referencia a las piezas léxicas (o ítems léxicos) y objetos sintácticos que configuran la numeración (o repertorio seleccionado de objetos que entran en una derivación sintáctica), O_s hace referencia a un objeto sintáctico cualquiera de la serie generada (que puede tomar n-valores: si n = 1, entonces O_s = X; si n = 2, entonces O_s = X, Y, y así sucesivamente) y $M(O_s)$ hace referencia a un objeto sintáctico nuevo generado a partir de O_s mediante la aplicación de M ( ) –esto es, *Merge*–.

Como se ha indicado en otros trabajos (Mota, 2013, 2014, 2015), este procedimiento genera/define recursivamente objetos sintácticos, independientemente de su organización interna –esto es, presenten o no auto-inclusión–.

Así, sea la numeración N = {*Juan, canta, muchas, canciones*}, lo que *Merge* hace es tomar dos ítems léxicos (definida aquí como una operación binaria) 'muchas' y 'canciones' y forma el objeto sintáctico {muchas, canciones} = {X,Y}. Mediante otra aplicación, *Merge* genera un nuevo objeto sintáctico {canta, {muchas, canciones}}, formado a partir de un objeto previamente computado ({muchas, canciones}); por último, *Merge* genera {Juan, {canta, {muchas, canciones}}}. Así, y de manera esquemática, se puede ver que lo que *Merge* hace (esto es, generar/definir recursivamente objetos sintácticos, independientemente de su estructura interna, tal como se muestra en la definición formal) queda expresado mediante la siguiente serie: O_s, $M'O_s$, $M'M'O_s$…; en el ejemplo: $O_{s\{muchas,\ canciones\}}$; $M_{canta}{}'Os_{\{muchas,\ canciones\}}$; $M_{Juan}{}'M_{\text{canta}}{}'Os_{\{muchas,\ canciones\}}$. Esto puede definirse recursivamente de manera formal como sigue (Mota, 2013, nota 2; 2014, p. 38; 2015, p. 167):

$M^{(0)}{}'(O_s) = O_s$, Def.
$M^{(n)}{}'(O_s) = M'M^{(n-1)}{}'(O_s)$ Def.

Pero el procedimiento empleado por Chomsky y definido por medio de este sistema de ecuaciones recursivas, no sólo genera, como acaba de hacerse patente, expresiones sin auto-inclusión (como *Juan canta muchas canciones*), sino que también genera estructuras con auto-inclusión como *el ratón que el gato que el perro perseguía mordió corría*, la cual exhibe inclusión central y, además, auto-inclusión, dado que una cláusula de relativo se incluye dentro de otra del mismo tipo. Antes de continuar, conviene advertir que según lo que se acaba de exponer, un dominio sin auto-inclusión también puede ser definido por recursión y, por tanto, como se mostró en el apartado anterior, la auto-inclusión no justifica la recursión y, por ello, ambos conceptos no deberían identificarse.

Dicho esto, la descripción de cómo genera *Merge* la oración de relativo especificativa *El ratón que el gato que el perro perseguía mordió corría* sería, en términos

generales y de manera algo más compleja que en el caso anterior, como sigue:

Ensamble de *el* y *perro*: {el, perro}
$[O_{s\{\text{el, perro}\}}]$

Ensamble de *el perro* y *perseguía*: {{el, perro}, perseguía}
$[O_{s\{\{\text{el, perro}\},\ \text{perseguía}\}}]$

Ensamble de *que* y *el perro perseguía*: {que, {{el, perro}, perseguía}}
$[M_{\text{que}}\,'O_{s\{\{\text{el, perro}\},\ \text{perseguía}\}}]$

Ensamble de *gato* y *que[…]*: {gato, {que, {{el, perro}, perseguía}}}
$[M_{\text{gato}}\,'M_{\text{que}}\,'O_{s\{\{\text{el, perro}\},\ \text{perseguía}\}}]$

Ensamble de *el* y *gato que[…]*: {el, {gato, {que, {{el, perro}, perseguía}}}}
$[M_{\text{el}}\,'M_{\text{gato}}\,'M_{\text{que}}\,'O_{s\{\{\text{el, perro}\},\ \text{perseguía}\}}]$

Ensamble de *el gato que[…]* y *mordió*: {{el, {gato, {que, {{el, perro}, perseguía}}}}, mordió}
$[M_{\text{el}}\,'M_{\text{gato}}\,'M_{\text{que}}\,'M_{\{\{\text{el, perro}\},\ \text{perseguía}\}}\,'O_{s\ \text{mordió}}]$

Ensamble de *que* y *el gato[…]*: {que, {{el, {gato, {que, {{el, perro}, perseguía}}}}, mordió}}
$[M_{\text{que}}\,'M_{\text{el}}\,'M_{\text{gato}}\,'M_{\text{que}}\,'M_{\{\{\text{el, perro}\},\ \text{perseguía}\}}\,'O_{s\ \text{mordió}}]$

Ensamble de *ratón* y *que el gato[…]*: {ratón, {que, {{el, {gato, {que, {{el, perro}, perseguía}}}}, mordió}}}
$[M_{\text{ratón}}\,'M_{\text{que}}\,'M_{\text{el}}\,'M_{\text{gato}}\,'M_{\text{que}}\,'M_{\{\{\text{el, perro}\},\ \text{perseguía}\}}\,'O_{s\ \text{mordió}}]$

Ensamble de *el* y *ratón que el gato[…]*: {el, {ratón, {que, {{el, {gato, {que, {{el, perro}, perseguía}}}}, mordió}}}}
$[M_{\text{el}}\,'M_{\text{ratón}}\,'M_{\text{que}}\,'M_{\text{el}}\,'M_{\text{gato}}\,'M_{\text{que}}\,'M_{\{\{\text{el, perro}\},\ \text{perseguía}\}}\,'O_{s\ \text{mordió}}]$

Ensamble de *el ratón[…]* y *corría*: {{el, {ratón, {que, {{el, {gato, {que, {{el, perro}, perseguía}}}}, mordió}}}}, corría}
$[M_{\text{el}}\,'M_{\text{ratón}}\,'M_{\text{que}}\,'M_{\text{el}}\,'M_{\text{gato}}\,'M_{\text{que}}\,'M_{\{\{\text{el, perro}\},\ \text{perseguía}\}}\,'M_{\text{mordió}}\,'O_{s\ \text{corría}}]$

Por otra parte, Chomsky (2007b, p. 6; 2008, p. 139) ha insistido en que el modo de operar (cómo procede o puede implementarse de manera abstracta) de este procedimiento generativo consiste en aplicar iterativamente la operación

Merge[8]. Así, la aplicación iterativa queda descrita mediante la siguiente serie, que muestra, en cada paso, las *n* veces que *Merge* se aplica de forma sucesiva, lo cual puede generar todo tipo de expresiones (con auto-inclusión o no) del lenguaje natural: $M^{(0)}$, $M^{(1)}$, $M^{(2)}$,…, $M^{(n)}$. Vemos, una vez más, que las estructuras con auto-inclusión no justifican la propiedad de la recursión.

Así, en el caso de la oración *Juan canta muchas canciones* (una oración que no presenta constituyentes dentro de constituyentes del mismo tipo y, por tanto, sin auto-inclusión a tal nivel), la aplicación iterativa es como sigue:

M_{muchas} '$O_{s\ \text{canciones}} = Os_{\{muchas,\ canciones\}}$ lo que equivale a {A,B} = C;
[M_{canta} '$Os_{\{muchas,\ canciones\}}$]= {C,D}= E;
M_{Juan} 'M_{canta} '$Os_{\{muchas,\ canciones\}}$= {E,F} = G
En donde la secuencia completa es:
{A,B} = C; {C,D}= E; {E,F} = G; que se resume en $M^{(3)}$'(A)

Esto indica que la operación *Merge* se ha aplicado tres veces sucesivamente (iterativamente) sobre su último resultado generado.

Por su parte, en el caso de la oración *el ratón que el gato que el perro perseguía mordió corría* (la cual presenta auto-inclusión) la generación iterativa sería como sigue (se da por hecho el ensamble de *el* y *perro*):

[$O_{s\{\text{el, perro}\},\ \text{perseguía}\}$]; {A,B} = C

[M_{que} '$O_{s\{\text{el, perro}\},\ \text{perseguía}\}$]; {C,D} = E

[M_{gato} 'M_{que} '$O_{s\{\text{el, perro}\},\ \text{perseguía}\}$]; {E,F} = G

[M_{el} 'M_{gato} 'M_{que} '$O_{s\{\text{el, perro}\},\ \text{perseguía}\}$]; {G,H} = I

[M_{el} 'M_{gato} 'M_{que} '$M_{\{\text{el, perro}\},\ \text{perseguía}\}$ '$O_{s\ \text{mordió}}$] ; {I,J} = K

[M_{que} 'M_{el} 'M_{gato} 'M_{que} '$M_{\{\text{el, perro}\},\ \text{perseguía}\}$ '$O_{s\ \text{mordió}}$] ; {K,L} = M

[$M_{\text{ratón}}$ 'M_{que} 'M_{el} 'M_{gato} 'M_{que} '$M_{\{\text{el, perro}\},\ \text{perseguía}\}$ '$O_{s\ \text{mordió}}$]]; {M,N} = P

[8] Es interesante apuntar cómo Wittgenstein distingue entre un procedimiento recursivo e iterativo. Así, siguiendo a Wittgenstein (1974, 32), éste dice que es claro que debe haber una "prueba" iterativa que "transmita la idea de que así debe ocurrir con todos los números"; esto es, una regla que concuerda con la inducción: $f(1) = a+b$; $f(1) \times (a+b) = (a+b)^2 = f(2)$; $(a+b)^n = f(n)$; $f(n) \times (a+b) = f(n+1)$. Así "una vez que se tiene la inducción, todo ha terminado" (Ibíd.). De este modo, la inducción matemática puede expresarse mediante una prueba iterativa.

$[M_{el}\,'M_{ratón}\,'M_{que}\,'M_{el}\,'M_{gato}\,'M_{que}\,'M_{\{\{el,\ perro\},\ persegu\acute{i}a\}}\,'O_{s\ mordi\acute{o}}]$; {P,Q} = R

$[M_{el}\,'M_{ratón}\,'M_{que}\,'M_{el}\,'M_{gato}\,'M_{que}\,'M_{\{\{el,\ perro\},\ persegu\acute{i}a\}}\,'M_{mordi\acute{o}}\,'O_{s\ corr\acute{i}a}]$; {R,S} = T

Cuya secuencia completa es:

{A,B} = C; {C,D} = E; {E,F} = G; {G,H} = I; {I,J} = K; {K,L} = M; {M,N} = P; {P,Q} = R; {R,S} = T; que se resume en $M^{(9)}$('A)

En este caso, *Merge* se aplica nueve veces (dando por hecho el ensamble de *el* y *perro*) de manera sucesiva sobre el último resultado producido, generando iterativamente una expresión jerárquicamente estructurada como la empleada en el ejemplo.

Como en el caso expuesto en la sección precedente, encontramos que la recursión es independiente de la auto-inclusión, en tanto en cuanto se puede definir por recursión todo objeto sintáctico con independencia de su organización interna. Asimismo, se puede implementar (de manera abstracta) el procedimiento generativo/computacional propuesto por Chomsky de manera iterativa, tal como ha quedado expuesto aquí, de tal modo que procediendo iterativamente se pueden generar estructuras sin y con auto-inclusión. En suma, la auto-inclusión no es una propiedad central de la recursión, como se ha venido proponiendo de forma reiterada[9].

4. ALGUNAS CONSECUENCIAS DE ESTA PROPUESTA

En este apartado se busca exponer algunas de las consecuencias que se pueden derivar al asumir esta propuesta, la cual adopta un punto de vista wittgensteiniano. En primer lugar, se va a analizar si la eventual existencia de lenguas sin estructuras recursivas (cf. Everett, 2005, 2009) no sería un argumento en contra de la recursividad como parte de la Gramática Universal (GU) (o como propiedad esencial de la facultad estrecha del lenguaje). Según el análisis realizado, que una lengua, como puede ser el Pirahã, no exhiba estructuras recursivas, esto es, estructuras con auto-inclusión, no es un argumento en contra de la recursión como parte de la GU, o como propiedad de la facultad del lenguaje (en sentido estrecho). Lo que el análisis precisamente muestra es que tales estructuras son, *simplemente*, estructuras con auto-inclusión, y, por tanto, no tienen que ver con la propiedad de la recursión, en tanto en cuanto esta última caracteriza el procedimiento

[9] Un trabajo muy reciente que sostiene que la auto-inclusión sí es una propiedad central de la recursión es el de Vicari y Adenzato (2014).

generativo/computacional. El problema surge –a juicio de quien escribe– de la confusión entre propiedades y niveles de análisis. Nótese que aunque se reconozca un esquema universal aplicable a las estructuras, a saber, el conocido esquema de [especificador-[núcleo-complemento]], el análisis no cambia nada, pues la recursión sigue aplicándose al procedimiento, mientras que tal esquema se aplica a las estructuras. Por tanto, desde el análisis efectuado no se deriva ninguna confusión a este respecto.

Con todo, esto puede conllevar un problema: qué hecho en el mundo podría contar como argumento en contra de la recursividad; en otras palabras, cuáles son las condiciones de falsabilidad de la idea de recursividad, si es que las hay. Esta cuestión, sumamente interesante, está lejos de presentar un problema real. Así, lo primero que se debe plantear es hasta qué punto el problema de la recursión es un problema empírico. Desde la perspectiva que se defiende en este trabajo, no se trata de un problema eminentemente empírico, sino matemático o formal. En este sentido, se considera que Kenny (1990) expuso la cuestión de manera magistral al proponer que la mente es "la capacidad de adquirir habilidades intelectuales" (Kenny, 1990, p. 204). Para este autor, hay que distinguir entre poseedores y vehículos. El poseedor de la habilidad lingüística es un ser humano y el vehículo "es la parte de su poseedor en virtud de la cual éste es capaz de ejercer la habilidad… algo concreto y más o menos tangible" (*op.cit.*, p. 205). Se puede hacer una distinción entre las personas (el poseedor), la mente (conjunto de capacidades) y el cerebro (el vehículo de tales capacidades). Evidentemente, las personas y sus cerebros son objetos físicos, pero no así la mente, que es un conjunto de capacidades. Conviene tener presente también que tal distinción no es una distinción metafísica/ontológica, sino conceptual, en virtud de la cual se definen 'mente' y 'vehículo'. Para ilustrar la cuestión, Kenny propone que pensemos en una calculadora como analogía de lo que se acaba de exponer: el fisiólogo estaría a la par con el ingeniero electrónico, mientras que el psicólogo está en una posición análoga a la del matemático, pues su papel consiste en intentar descubrir el algoritmo que usa la calculadora (1990, p. 206). La calculadora tiene la habilidad de computar un algoritmo y su análisis exige una investigación matemática, más que electrónica. El aparataje electrónico sería el vehículo.

Sin embargo, si se atiende al *naturalismo metodológico* que defiende Chomsky, el lenguaje, entendido como un órgano *mental* o un sistema biológico, ha de ser considerado como un objeto real, esto es, como el resto de objetos del mundo, y su estudio ha de ser abordado desde una perspectiva naturalista, esto es, como las ciencias naturales estudian otros objetos del mundo. Bajo este naturalismo metodológico, tal y como se acaba de exponer, Chomsky hace una aproximación a la mente bajo la cual considera el lenguaje como un elemento del mundo natural (1995b, p. 1). Chomsky subraya aquí que usa el término 'mente' y 'mental' sin ninguna connotación metafísica, estando a la par de términos como 'químico',

'óptico' o 'eléctrico'. Tales términos seleccionan ciertos aspectos del mundo como foco de investigación, entendiendo por 'mente' los aspectos mentales del mundo.

Desde la perspectiva biolingüística, nos dice Chomsky (2006), el lenguaje es un componente de la mente, entendiendo 'mente' en el sentido de los científicos del siglo XVIII, de tal manera que, no siendo posible postular un problema mente/cuerpo coherente, sólo se pueden considerar los aspectos del mundo denominados mentales como el resultado de una organización orgánica tal como la del cerebro (p. 173; véase también 2005, p. 2).

Estas consideraciones generales han sido algo más precisadas en un interesante trabajo recientemente publicado. En dicho trabajo, Berwick, Friederici, Chomsky y Bolhuis (2013) señalan que el lenguaje humano está asentado sobre un mecanismo computacional particular realizado neurológicamente (p. 90). En este primer apunte, es interesante notar que el mecanismo computacional no es sólo una herramienta construida para caracterizar el lenguaje de manera abstracta, sino que se postula que tal procedimiento mecánico finito está realizado neurológicamente. Es decir, tal procedimiento está a la par que, por ejemplo, las neuronas; son objetos del mundo. Por decirlo de otra manera, dicho procedimiento no sólo es una clase especial de funciones recursivas (ver *supra*), sino que también es un objeto del mundo.

Como se señaló unas líneas más arriba, el procedimiento mecánico finito basado en *Merge* genera objetos sintácticos que son interpretados por dos sistemas: el sensorio-motor (encargado entre otras cosas de la externalización) y el intencional-conceptual (que hace referencia, entre otras cuestiones, al pensamiento y a la planificación de la acción) (Chomsky, 2011, p. 269). Obviamente, en el plano neurológico, tal mecanismo generativo/computacional debe diferenciarse de ambos sistemas de interfaz. Berwick et al. (2013, p. 93) presentan diferentes regiones identificadas tanto con el mecanismo computacional como con los sistemas de interfaz. Así, dicho mecanismo está sustentado (o realizado neurológicamente) por el área de Brodmann 44 y el córtex temporal superior posterior. El sistema sensorio-motor, por su parte, estaría sustentado por el córtex premotor y el córtex temporal superior. Finalmente, el procesamiento semántico (propio del sistema conceptual-intencional) está sustentado por el área de Brodmann 45 y por el córtex frontal inferior, así como por porciones del córtex temporal. Por tanto, cada uno de estos sistemas consisten en las mencionadas regiones particulares del cerebro, conectadas vía tractos específicos (véase Berwick et al., 2013 para más detalles).

El estudio del lenguaje en este nivel neurológico es, siguiendo a Kenny, el estudio del vehículo, del cerebro. El mecanismo computacional (cuyo estudio apropiado es una investigación matemática) no consiste en estas o aquellas regiones cerebrales. Tal procedimiento es una construcción matemática que no supone el descubrimiento de ningún objeto mental del mundo natural. El estudio del cere-

bro, por su parte, sí es el estudio de un objeto del mundo natural, susceptible de una metodología naturalista. Eso sí, el estudio se realiza en otro nivel de análisis.

A modo de analogía, se puede pensar que en una ciencia como la física, que describe y predice con cierto éxito los acontecimientos de la realidad circundante, ha de considerarse que si una teoría es verdadera, entonces *deben* existir las entidades referidas por los términos de la teoría. De ahí que, de acuerdo con Putnam (1972, p. 57), algunos autores, como por ejemplo Quine, se hayan comprometido con la existencia de entidades matemáticas, dado que la cuantificación sobre tales entidades (i.e., números, funciones, conjuntos) es indispensable tanto en la ciencia física como en la ciencia formal. Por otro lado, el paso de la cuantificación sobre objetos matemáticos a la afirmación de su existencia (un paso nada obvio) es un ejemplo de la aplicación del criterio ontológico de Quine (véase Alemán, 2011 para más detalles).

Así pues, una cosa es que se construya un sistema formal, un cálculo, y se lleve a cabo un análisis matemático del mismo, y otra cosa completamente diferente es que (I) se lo postule como un objeto de una realidad (empírica o platónica), y (II) se entienda que ambos niveles de análisis, a saber, el lógico-matemático y el empírico, en este caso, no sólo están al mismo nivel, sino que son lo mismo; esto es, las entidades físicas y matemáticas existen en la misma realidad espacio-temporal y, por tanto, se las aborda mediante una metodología naturalista. Esta concepción realista del mecanismo formal propuesto por Chomsky parece estar detrás de la idea de que dicho mecanismo es un objeto del mundo natural.

Sin embargo, una cosa es decir que el lenguaje es un sistema biológico y otra muy diferente decir que la caracterización abstracta que se deriva de la formalización planteada por Chomsky sea un objeto real del mundo natural; es decir, que tal formalización describa un objeto del mundo natural. Tal formalismo es, en cierto sentido, autónomo con respecto a cualquier realidad, en el sentido de que no es descriptivo de la misma. En este sentido, en el momento en el que se construye un formalismo se están constituyendo tanto los signos lógico-matemáticos empleados en éste como las reglas para su uso dentro del sistema (por ejemplo, definidos unos axiomas, un sistema formal o procedimiento mecánico finito puede calcular en un número finito de pasos otras verdades lógicas mediante la aplicación de operaciones siguiendo un conjunto de reglas, pudiendo ordenarse tales verdades en series; cf. supra). De este modo, los objetos mentales como el mecanismo generativo/computacional propuesto por Chomsky no se descubren, sino que se crean, se construyen y se constituyen mediante las reglas gramaticales (en sentido wittgensteiniano). Por ello, el carácter esencial del sistema formal definido por Chomsky no es su carácter descriptivo, sino constitutivo o constructivo, como se acaba de indicar.

Adicionalmente, surge la duda de cuáles serían las condiciones en que podría asignarse validez o realidad psicológica a la recursividad. Esta propuesta es com-

patible con un enfoque no psicológico del lenguaje, esto es, con la tesis de que el lenguaje no es un *objeto* mental. En este caso, ¿cuál sería el estatus de la recursividad? Relacionado con esto último, conviene recordar que una regla gramatical no niega ni asevera nada y, por tanto, no describe ninguna realidad. Así, por ejemplo, se podría distinguir entre:

(a) La expresión "2+2" se puede intercambiar con la expresión "(2+1)+1".
(b) El sujeto S usa la regla 'R'.
(c) El sujeto S usa la regla 'R''.

(a) no es un enunciado empírico y, por tanto, las propiedades constituidas por dicha regla gramatical no dependen de ninguna realidad; son autónomas frente a ésta y se constituyen por la regla y, por tanto, dentro del sistema. Por su parte, los enunciados (b) y (c) son enunciados empíricos y, por tanto, lo que se confirma o desconfirma son tales enunciados, no las reglas 'R' o 'R''. Dichas reglas, así como (a), no *dicen* nada, no describen ninguna realidad.

Continuando con el análisis mostrado unas líneas más arriba, la recursión no tiene realidad psicológica en el sentido de que su investigación es matemática y, por tanto, no hay que postular la existencia de un mundo matemático poblado de entidades independientes de nosotros. Dicho de otra manera, la recursión no es una propiedad surgida de la relación entre entidades matemáticas existentes independientemente. Como se indicó antes, la investigación formal del procedimiento mecánico finito propuesto por Chomsky no implica necesariamente ningún tipo de criterio ontológico por medio del cual sea necesario postular la existencia de determinadas entidades. Por el contrario, el carácter constitutivo o constructivo del sistema generativo/computacional pone de manifiesto que es una herramienta o instrumento útil para realizar una caracterización abstracta de la facultad del lenguaje, proponiendo un procedimiento que permite computar en un número finito de pasos un resultado u *output* lingüístico dado un *input* lingüístico. Pero, hay que insistir, que su investigación no es psicológica, si por 'investigación psicológica' se entiende una investigación experimental, pues tal procedimiento no es un descubrimiento del mundo empírico (ni, por supuesto, es un descubrimiento de un mundo platónico). Antes bien, es una creación o invención matemática.

De esto se desprende que el estatus de la recursión no es el de una propiedad del mundo empírico o platónico, sino una propiedad interna a un cálculo creado o inventado. Dicho de otra manera, es una propiedad de una técnica diseñada para definir y calcular y, por tanto, operar de una determinada manera. Entendida así, un experimento no debería proponerse descubrirla, sino aclarar cómo se aplica para caracterizar un comportamiento lingüístico. En suma, la recursión no es un objeto que descubrir en el mundo, sino una herramienta que, en el mejor de los casos, se puede aplicar.

5. CONCLUSIONES

En el presente trabajo se ha tratado de poner de relieve que la literatura especializada confunde de manera recurrente la recursión con la auto-inclusión. Tal como se analizó, el sentido original y primario de 'recursión', entendida como propiedad de una regla, no puede identificarse con la auto-inclusión y esta última no justifica la primera. Una vez que Chomsky tomó el concepto de recursión de la Teoría de la Computabilidad y la Lógica Matemática, y propuso que tal propiedad era constitutiva de su procedimiento generativo, muchos autores comenzaron a aplicar la recursión como una propiedad de las estructuras. Sin embargo, tal como se espera haber demostrado, la recursión es una propiedad de una definición, de una regla, y la auto-inclusión, propiedad que caracteriza a las estructuras, no justifica tampoco en la Ciencia Cognitiva la propiedad de la recursión, siendo ambas propiedades claramente independientes.

REFERENCIAS

Alemán, A. (2011). *Lógica, matemáticas y realidad*. Madrid: Tecnos.

Arsenijević, B. & Hinzen, W. (2012). On the abscence of X-within-X recursion in human grammar, *Linguistic Inquiry*, 43, 423-440.

Berwick, R., Friederici, A.D., Chomsky, N. & Bolhuis, J. (2013). Evolution, brain, and the nature of language, *Trends in Cognitive Science*, 17, 89-98.

Boolos, G. (1971). The iterative conception of set, *The Journal of Philosophy*, 68, 215-231.

Boolos, G. & Jeffrey, R. (1974). *Computability and logic*. Cambridge: Cambridge University Press.

Chomsky, N. (1955/1975). *The logical structure of linguistic theory*. Nueva York: Plenum Press.

Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control*, 2, 137-167.

Chomsky, N. (1965). *Aspects of the theory of syntax*. Cambridge. MA: MIT Press.

Chomsky, N. (1980). *Rules and representations*. Nueva York: Columbia University Press.

Chomsky, N. (1995a). *The minimalist program*. Cambridge. MA: MIT Press.

Chomsky, N. (1995b). Language and nature, *Mind*, 104, 1-61.

Chomsky, N. (2005). Three factors in language design, *Linguistic Inquiry*, 36, 1-22.

Chomsky, N. (2006). *Language and mind*. Cambridge: Cambridge University Press.

Chomsky, N. (2007a). Of minds and language, *Biolinguistics*, 1, 9-27.

Chomsky, N. (2007b). Approaching UG from below. En U. Sauerland & H. M. Gärtner (Eds.), *Interfaces + Recursion = Language?* (pp. 1-30). Berlin: Mouton.

Chomsky, N. (2008). On phases. En R. Freidin, C. Otero, & M. L. Zubizarreta (Eds.), *Foundational issues in linguistic theory* (pp. 133-166). Cambridge. MA: MIT Press.

Chomsky, N. (2010). Some simple evo-devo theses: How true might they be for language? En R. Larson, V. Déprez & H. Yamakido (Eds.), *The evolution of human language* (pp. 45-62). Cambridge: Cambridge University Press.

Chomsky, N. (2011). Language and other cognitive systems. What is special about language?. *Language Learning and Development,* 7, 263-278.

Chomsky, N. (2012). Some core contested concepts. *Actas de la 25th Annual CUNY Conference on Human Sentence Processing* (pp. 1-18).

Chomsky, N. & Miller, G. (1963). Introduction to the formal analysis of natural languages. En R.D. Luce, R.R. Bush & E. Galanter (Eds.), *Handbook of mathematical psychology*, Vol. II (pp. 269-322). New York: John Wiley.

Corballis, M. C. (2007). Recursion, language and starlings, *Cognitive Science*, 31, 69-704.

Cutland, N. (1980). *Computability: an introduction to recursive function theory*. Cambridge: Cambridge University Press.

Davis, M. (1982). Why Gödel didn't have Church's Thesis. *Information and Control*, 54, 3-24.

Eguren, L. & Fernández Soriano, O. (2004). *Introducción a una sintaxis minimista*. Madrid: Gredos.

Epstein, R. & Carnielli, W. (1989). *Computability: computable functions, logic, and the foundations of mathematics*. Pacific Grove, CA: Wadsworth & Brooks/Cole.

Everett, D. (2005). Cultural constraints on grammar and cognition in Pirahã, *Current Anthropology*, 46(4), 621-646.

Everett, D. (2009). Pirahã culture and grammar: a response to some criticisms, *Language*, 85(2), 405-442.

Fitch, T. (2010). Three meanings of recursion: Key distinctions for biolinguistics. En R. Larson, V. Déprez & H. Yamakido (Eds.), *The evolution of human language* (pp. 73-90). Cambridge: Cambridge University Press.

Gödel, K. (1931). On formally undecidable propositions of the Principia Mathematica and related systems. I. In M. Davis (Ed.), *The undecidable* (pp. 4-38). New York: Raven Press.

Gödel, K. (1934). On undecidable propositions of formal mathematical systems. In M. Davis (Ed.), *The undecidable* (pp. 39-74). New York: Raven Press.

Hauser, M., Chomsky, N. & Fitch, T. (2002). The faculty of language: What is, who has it, and how did it evolve?, *Science*, 298, 1569-1579.

Jackendoff, R. (2011). What is the human language faculty? Two views, *Language*, 87, 586-624.

Karlsson, F. (2010). Syntactic recursion and iteration. En H. van der Hulst (Ed.), *Recursion and human language* (pp. 43-67), Berlín: De Gruyter Mouton.

Kenny, A. (1990). *El legado de Wittgenstein*. Madrid: Siglo XXI.

Kinsella, A. (2010). Was recursion the key step in the evolution of the human language faculty? En H. van der Hulst (Ed.), *Recursion and human language* (pp. 179-191) , Berlín: De Gruyter Mouton.

Kleene, S. C. (1938). On notation for ordinal numbers, *The Journal of Symbolic Logic*, 3, 150-5.

Kleene, S. C. (1943). Recursive predicates and quantifiers, *Transactions of the American Mathematical Society*, 53, 41-73.

Kleene, S. C. (1952). *Introduction to metamathematics*. Amsterdam: North-Holland Publishing.

Kleene, S. C. (2002). *Mathematical logic*. Mineola, NY: Dover Publications.

Luuk, E. & Luuk, H. (2011). The redundancy of recursion and infinity for natural language, *Cognitive Processing*, 12, 1-11.

Moro, A. (2008). *The boundaries of Babel*. Cambridge, MA: MIT Press.

Mota, S. (2013). La propiedad de la recursión en el "Tractatus Logico-Philosophicus" de Wittgenstein y su relación con la Teoría de la Computabilidad y la Lógica Matemática, 17, *Observaciones Filosóficas.* Disponible en http://www.obervacionesfilosoficas.net/lapropiedaddelarecursion.htm

Mota, S. (2014). La historia y la gramática de la recursión: una precisión desde la obra de Wittgenstein, *Pensamiento y Cultura*, 17, 20-48.

Mota, S. (2015). Sobre el concepto de recursión y sus usos, *Praxis Filosófica*, *40*, 153-181.

Pinker, S. & Jackendoff, R. (2005). The faculty of language: What's special about it?, *Cognition*, 95, 201-236.

Post, E. (1921). Introduction to a general theory of elementary propositions, *American Journal of Mathematics*, 43, 163-185.

Post, E. (1943). Formal reductions of the general combinatorial decision problem, *American Journal of Mathematics*, 65, 197-215.

Post, E. (1944). Recursively enumerable sets of positive integers and their decision problems. I. En M. Davis (Ed.), *The undecidable* (pp. 305-337). Nueva York: Raven Press.

Putnam, H. (1972). *Philosophy of logic*. Londres: George Allen and Unwin.

Skolem, T. (1923). The foundations of elementary arithmetic established by means of the recursive mode of thought, without the use of apparent variables ranging over infinite domains. En J. Van Heijenoort (Ed.), *From Frege to Gödel. A source book in mathematical logic, 1879-1931* (pp. 302-333). Cambridge, MA: Harvard University Press.

Soare, R. (1996). Computability and recursion, *The Bulletin of Symbolic Logic*, 2, 284-321.

Soare, R. (1999). The history and concept of computability. En E.R. Griffor (Ed.), *Handbook of computability theory* (pp. 3-36,). Amsterdam: North-Holland Publishing.

Soare, R. (2009). Turing oracles machines, online computing, and three displacements in computability theory, *Annals of Pure and Applied Logic*, 160, 368-399.

Tomalin, M. (2007). Reconsidering recursion in syntactic theory, *Lingua*, 117, 1784-1800.

Tomalin, M. (2011). Syntactic structures and recursive devices: A legacy of imprecision, *Journal of Logic, Language and Information*, 20, 297-315.

Vicari, G. & Adenzato, M. (2014). Is recursive language-specific? Evidence of recursive mechanisms in the structure of intentional action, *Consciousness and Cognition*, 26, 169-188.

Wittgenstein, L. (1974). *Philosophical grammar*. Oxford: Blackwell.

Wittgenstein, L. (1975). *Philosophical remarks*. Oxford: Blackwell.

ARTÍCULO 3

Mota, S. (2014). Sobre el anti-realismo de Wittgenstein y su aplicación al programa chomskiano, Metatheoria, 4. (En proceso de edición).

Sobre el anti-realismo de Wittgenstein y su aplicación al programa chomskiano*

On Wittgenstein's Anti-realism and its Application to the Chomskyan Program

Sergio Mota†*

Resumen

El objetivo principal de este trabajo es mostrar un mapa conceptual en el que situar a Wittgenstein dentro de las distintas concepciones acerca de la lógica y la matemática. Después analizaré cómo se puede aplicar su concepción de la matemática al formalismo. Por último, analizaré el procedimiento mecánico finito definido por Chomsky y su relación con las nociones de mente y de representación (mental) dentro del marco conceptual expuesto previamente.

Palabras clave: anti-realismo - Wittgenstein - Chomsky - representación mental - mente

Abstract

The main goal of this paper is to present a conceptual framework in which we can situate Wittgenstein's views and compare them with different conceptions about the nature of Logic and Mathematics. Then I will consider how his conception of Mathematics can be applied to the formalism. Finally, I will analyze the finite, mechanical procedure defined by Chomsky and its relation to the notions of mind and (mental) representation, within the conceptual framework previously exposed.

Keywords: anti-realism - Wittgenstein - Chomsky - mental representation - mind

†Departamento de Psicología Básica, Universidad Autónoma de Madrid, España. Para contactar al autor, por favor escriba a: sergio.mota.v@gmail.com.

* Mi más sincero agradecimiento a los profesores Anastasio Alemán y Noam Chomsky por su inestimable ayuda y disponibilidad en las discusiones de algunos de los puntos analizados en el presente artículo. Asimismo mi gratitud al profesor Carlos U. Moulines por su lectura pormenorizada y útiles comentarios a una versión previa.

* Recibido: 3 de Febrero de 2014. Aceptado con revisiones: 13 de Marzo de 2014.

I

A lo largo de la historia se han propuesto diferentes concepciones acerca de la naturaleza de la lógica y la matemática. El análisis que voy a presentar aquí le debe mucho al que ha presentado Alemán (2011) con anterioridad. Fundamentalmente, voy a seguir las líneas maestras por él esbozadas incluyendo algunas concepciones que no aborda directamente. En su libro *Lógica, matemáticas y realidad* (Alemán 2011) recoge una distinción general entre las diferentes concepciones de la lógica y la matemática y su relación con la realidad. Así, sugiere una primera división entre aquellas concepciones que son *descriptivistas* y aquellas que *son no-descriptivistas*.

Las descriptivistas hacen referencia a aquellas concepciones que entienden las expresiones lógico-matemáticas como descripciones de la realidad. Sostienen, por tanto, que tales expresiones dicen algo acerca de algún aspecto de la realidad que describen. De qué realidad hablemos nos lleva a una subdivisión: *Platonismo* y *Empirismo*. En mi opinión, en esta categoría habría que destacar también el *Logicismo*.

El *Platonismo* postula que la realidad que es descrita por los enunciados lógicos y matemáticos es una realidad ideal, abstracta, fuera de los límites espacio-temporales. Un autor de referencia es Gödel (véase, por ejemplo, Gödel 1944).

Frente al Platonismo encontramos el *Empirismo*. Para éste último, los enunciados lógico-matemáticos describen una realidad espacio-temporal, perceptible por los sentidos, siendo innecesario, por tanto, apelar a un tercer reino platónico. Dentro de esta concepción empirista, alternativa al empirismo lógico, encontramos a Maddy, Kitcher o Tymoczko, entre otros (recomiendo Alemán 2011, capítulos 1 y 2 para más detalles y referencias).

He propuesto más arriba el *Logicismo* como una concepción de la lógica y la matemática que cae dentro de los descriptivistas.[1] Para los logicistas, el aspecto

[1] En una comunicación personal con el profesor Moulines, me indicó que el logicismo, en efecto, no tiene por qué ser considerado como una forma de realismo. "En sí mismo considerado, el logicismo sólo sostiene que las matemáticas son reducibles a la lógica", y añade, "otra cuestión es que históricamente muchos logicistas hayan adoptado adicionalmente una posición realista". Considero esta caracterización correcta, y aquí,

fundamental es que creen que la matemática puede reducirse a la lógica. Sin embargo, logicistas reconocidos como Frege y Russell parecen sostener la existencia real de los objetos lógico matemáticos apelando a la posibilidad de usar variables ligadas para referirse indistintamente a entidades abstractas independientes de la mente y, por tanto, no construidas o inventadas; quizá descubiertas o, como dice Frege, captadas.

En relación con las concepciones no-descriptivistas, las cuales niegan que los enunciados lógico-matemáticos describan la realidad, esto es, nieguen o aseveren nada, se encuentran los constructivistas. Podríamos decir que dentro esta categoría se encuentran los *Intuicionistas*, *Formalistas* y *Convencionalistas*. Así, los primeros (al igual que el conceptualismo) sostienen que las construcciones lógico-matemáticas son, en todo caso, productos del ejercicio intelectual, construcciones. Es decir, los objetos lógico-matemáticos no se descubren, sino que se inventan. Para una concisa presentación del *Intuicionismo* véase Kleene (1952, pp. 46-53).

Por su parte, los formalistas, encabezados por Hilbert, consideran, en alguna medida, insatisfactorios tanto el logicismo como el intuicionismo. Quine (1980, p. 15), argumenta tal insatisfacción apelando a dos razonamientos opuestos. En el primero de ellos, concede que formalismo (relacionado con el nominalismo) e intuicionismo atacan conjuntamente el recurso de los logicistas de los universales (entidades abstractas que son descubiertas e independientes de la mente). En el segundo de ellos, el formalista se pone del lado de los logicistas y ambos atacan conjuntamente a los intuicionistas al oponerse a las restricciones específicas de la matemática intuicionista. Para una presentación del *Formalismo*, recomiendo Kleene (1952, pp. 53-65).

El *Formalismo* es, tal como indica Alemán (2011, p. 19), el antecedente histórico del *Convencionalismo* y se aprecian ciertamente semejanzas entre ambas concepciones. Así, los convencionalistas tampoco encuentran razones suficientes para desestimar los sistemas formales de la matemática clásica construidos de manera diferente a como lo hace la matemática intuicionista. De este modo, se aceptan diferentes sistemas formales definidos con base en diferentes reglas para

siguiendo a Quine (1980), hago referencia a logicistas como Frege y Russell, los cuales parecen haber adoptado una posición realista.

el manejo de los signos lógico-matemáticos; se concibe como un juego de notaciones formales. Dicho de otro modo, las expresiones o enunciados de la lógica y de la matemática son reglas, reglas gramaticales, como diría Wittgenstein. De este modo, se pueden investigar y, de hecho, esto ya se ha realizado, propiedades como la "completud" o la "decidibilidad" (véase Gödel 1931, Church 1936, Turing 1937, Post 1943). Finalmente, es interesante apuntar que aunque la filosofía de la matemática de Wittgenstein se sitúe dentro del convencionalismo, este autor no pareció interesarse por los fundamentos de la matemática, como sí hicieron los formalistas, algo que a mi juicio es relevante (véase Monk 1990).

En la siguiente sección, desarrollaré en mayor detalle la posición anti-realista de Wittgenstein a lo largo de su obra. En la tercera sección discutiré bajo el prisma wittgensteiniano algunas de las propuestas más importantes dentro de la concepción formalista, proponiendo que todos los sistemas formales son objetos lógico-matemáticos construidos y, por tanto, no aluden a entidades independientes descubiertas por el lógico o el matemático. En la cuarta sección abordaré algunas de las implicaciones del análisis previo para la discusión sobre la naturaleza de lo mental, fundamentalmente dentro del programa chomskiano. Finalmente, presentaré las conclusiones.

II

En esta sección voy a presentar el anti-realismo que caracterizó la obra de Wittgenstein, desde la aparición del *Tractatus Logico-Philosophicus* (1922), pasando por las *Philosophical Remarks* ([1975a] 1997) y *Philosophical Grammar* ([1974] 1992), hasta llegar a las *Philosophical investigations* (1958), centrándose en esta última etapa, sobre todo, en la noción de regla gramatical. La idea de que la matemática está constituida por reglas gramaticales también es fundamental en sus *Remarks on the Foundations of Mathematics* (1978). Asimismo, defenderé que la lógica y la matemática no describen hechos, sino que son sistemas construidos, creados o inventados (uso estos términos de manera intercambiable).

En lo referente al *Tractatus*, hay diferentes interpretaciones sobre el anti-realismo de Wittgenstein, alternativa a la interpretación empirista que de tal

obra intentaron establecer los positivistas. Así, Pinto (2002) defiende el anti-platonismo presente en el *Tractatus*, aunque a mi juicio, los ejemplos que expone para ello no parecen ser los que mejor muestran tal posición, y en todo caso no son los únicos. Así, Pinto (2002) fundamenta el anti-platonismo en la filosofía de la matemática del *Tractatus* apelando a la siguiente proposición 6.211, que dice:

> [E]n la vida, una proposición matemática no es jamás lo que necesitamos; más bien utilizamos las proposiciones matemáticas *sólo* para inferir, de proposiciones que no pertenecen a la matemática, otras que tampoco pertenecen a la matemática (cursivas en el original, Pinto 2002, p. 138).

Sin embargo, esta proposición no nos permite establecer por sí misma el anti-platonismo, ya que bien podría servir para establecer una posición anti-empirista de la matemática. Esa proposición quiere decir que utilizamos las reglas matemáticas para inferir de los enunciados 'tengo dos piedras' y 'tengo dos piedras' el enunciado 'tengo cuatro piedras'. Enunciados empíricos, en todo caso. Y es que esta postura anti-descriptivista en el *Tractatus* se refleja en que "las proposiciones de la matemática no expresan pensamiento alguno" (Pinto 2002, proposición 6.21, p. 138).

Una muestra más acertada, en mi opinión, del anti-platonismo del *Tractatus*, se evidencia en la proposición 6.232: "Frege dice que ambas expresiones [entiéndase '1+1+1+1' y '(1+1)+(1+1)'] tienen la misma referencia, pero diferente sentido" y continúa "[p]ero lo esencial de una ecuación es que no es necesaria para mostrar que las dos expresiones que combina el signo de igualdad tienen el mismo significado, pues esto puede desprenderse de las dos expresiones mismas". Es decir, Wittgenstein está negando que expresiones como '1+1+1+1' tengan una referencia, una entidad ideal a la que se refiere o que denota. Al mismo tiempo, está diciendo que tales expresiones no *dicen* que tengan el mismo significado, sino que eso se *muestra* en los signos; la ecuación sería la regla explícita que muestra tal igualdad.

En lo referente al dominio de la lógica, Frege sostuvo que la referencia de una función es su valor de verdad, esto es: lo Verdadero o lo Falso (Frege 1892). Esta concepción es también atacada por Wittgenstein en el *Tractatus*. "Mi idea fundamental es que las constantes lógicas no actúan como representantes de

nada" (4.0312); esto es, no son nombres cuyas referencias sean lo Verdadero o lo Falso. Esto es especialmente relevante en el caso de tautologías –o, siguiendo a Wittgenstein, "las proposiciones de la lógica" (6.1)– y contradicciones, pues ambas carecen de sentido, no expresan pensamiento alguno.

Por otro lado, como muy bien ha señalado Ruíz Abánades (2009), los nombres propios empleados en el *Tractatus*, en un sentido lógico, no son, como asumieron los positivistas, términos observacionales referentes a propiedades observables. Podemos decir, por tanto, que los nombres en este sentido lógico (formal) del *Tractatus* no representan propiedades observacionales. Éste párrafo y el anterior muestran claramente el anti-realismo expuesto por Wittgenstein en el Tractatus.

Con respecto a la noción de objeto (lógico o matemático) en el *Tractatus*, Wittgenstein muestra el uso que le da dentro de una concepción no-descriptivista (y no sólo anti-platonista). En 4.1272, Wittgenstein nos indica que "el nombre variable 'x' es el signo propiamente dicho del pseudoconcepto [o concepto formal] objeto" (en cursivas en el original). Y continúa "[s]iempre que la palabra 'objeto'... se usa correctamente, se expresa en las notaciones conceptuales mediante un nombre variable"; así, pone como ejemplo que la proposición "Hay dos objetos tales que..." se expresa en tal notación como "($\exists x,y$)...". "De este modo uno no podría decir, por ejemplo, 'Hay objetos' del mismo modo en que podría decir 'Hay libros'", por la sencilla razón de que lo primero no es una cuestión acerca de lo que hay, sino, sencillamente, conceptual. Nos dice Wittgenstein: "[l]a pregunta por la existencia de un concepto formal es un sinsentido. Pues ninguna proposición puede responder a tal pregunta" (4.1274), dado que no es una cuestión consistente en comparar una proposición con la realidad (empírica o platónica). Los objetos son, sencillamente, construcciones que pertenecen al simbolismo y por ello, ya se muestran en la notación.

Unida a la noción de objeto está, como he mostrado, la de concepto formal. En 4.12721, Wittgenstein indica que "[u]n concepto formal está ya dado en cuanto se da un objeto que cae bajo él" (cf. 4.126; 4.127; 4.1271). Conceptos formales (variables) son, por ejemplo, la forma general o término general de una serie de formas (cf. 4.1273). Tal forma general (o variable) queda expresada así:

$[a, x, O'x]$.[2] Tal expresión muestra una regla para definir series de formas y puede ser expresada por un sistema de ecuaciones recursivas (Mota 2013, 2014):[3]

$$O^{(0)}{}'(a) = a,$$
$$O^{(n)}{}'(a) = O'O^{(n-1)}{}'(a).$$

La recursión caracteriza la definición de cada término y, por tanto, de la serie como un todo (esto se corresponde con el nivel de análisis referente a qué hace un procedimiento tal; véase Mota 2013, 2014), mientras que la iteración caracteriza cómo se procede para generar cada término; a saber, aplicando en sucesión la operación O' sobre el último valor generado, lo que permite generar series como a, O'a, O'O'a,... y así sucesivamente (Mota 2013, 2014).

Lo mismo vale para los términos generales de una proposición '$[\tilde{p}, \tilde{\xi}, N(\tilde{\xi})]$' y de un número natural '$[\Omega^0{}'x, \Omega^{v-1}{}'x, \Omega^{v}{}'x] = [0, \xi, \xi + 1]$'.

La definición recursiva del procedimiento mecánico finito que nos proporciona Wittgenstein para generar toda proposición (o función de verdad) y decidir en un número finito de pasos si es o no una tautología es como sigue (Mota 2013, 2014):

$$N^{(0)}{}'(\tilde{\xi}) = \tilde{\xi} \text{ Def.},$$
$$N^{(n)}{}'(\tilde{\xi}) = N'N^{(n-1)}{}'(\tilde{\xi}) \text{ Def.}$$

Como el propio Wittgenstein señala en 5.502, "$N(\tilde{\xi})$ es la negación de todos los valores de la variable proposicional ξ". Por tanto, como indica en 5.51, si $\xi = p$, entonces $N(\tilde{\xi})$ = (FVFV) (p). Por otra parte, si $\xi = p,q$, entonces $N(\tilde{\xi})$ = (FFFV) (p,q). Así, la operación $N(\tilde{\xi})$ es equivalente a la operación de la negación conjunta, la cual se representa por medio de la barra de Sheffer (Mounce 1981). El procedimiento que arriba defino puede generar proposiciones como las que

[2] En 5.2522, explica en qué consiste esta notación. Así, "[e]l primer término...es el comienzo de la serie de formas, el segundo es la forma de un término x cualquiera de la serie y el tercero la forma de aquel término de la serie que sigue inmediatamente a x".

[3] Por 'recursión' hay que entender el término definido dentro de la Lógica Matemática, y más concretamente, dentro de una rama especializada de ésta, la teoría de la computabilidad. Siguiendo a Soare (1996, 2009), 'recursión' se predica del método o reglas para definir funciones y predicados. Así, una función está definida en sentido técnico por recursión cuando para definirla para un argumento x hacemos uso de sus propios valores previamente computados para argumentos menores que x; pudiendo emplearse también funciones previamente definidas (cf. Cutland 1980, Kleene 1943). Este es el significado original y primario de 'recursión' (véase Soare 1996, 2009, para más detalles).

Pinto (2002, p. 142) propone como ejemplo: [(p, q); (ξ, ζ); $N(N(N(\xi, \xi), \zeta), N(N(\xi, \xi), \zeta))$]. Una expresión tal se puede simplificar poniendo '$N(p, q)$' en lugar de '$N(N(\xi, \xi), \zeta)$', en donde '$N(\xi, \xi)$' está por '$p_{(FVFV)}$' y 'ζ' está por '$q_{(VVFF)}$'. Con esto tenemos '$N[N(p, q), N(p, q)]$' que equivale a '$N(N(p_{(FVFV)}, q_{(VVFF)}))$', puesto que es la negación conjunta de '$N(p_{(FVFV)}, q_{(VVFF)})$' consigo misma. Una definición recursiva de tal expresión es: $N^{(2)}{}'(p, q) = N'N^{(1)}{}'(p, q)$. En el caso que acabo de analizar, sería suficiente poner '$(VVFV)$ (p, q)' y el procedimiento mecánico finito arriba definido permite generar (y decidir en un número finito de pasos) otras proposiciones a partir de ella (cf. 6.002). Esto muestra, por otro lado, el *finitismo* que, en general, caracteriza la obra de Wittgenstein.

De hecho sería posible establecer el siguiente esquema, que completa a 5.101:

$(VVVV)$ (p, q)
$(FVVV)$ (p, q)
$(VFVV)$ (p, q)
$(VVFV)$ (p, q)
$(VVVF)$ (p, q)
$(FFVV)$ (p, q)
$(FVFV)$ (p, q)
$(FVVF)$ (p, q)
$(VFFV)$ (p, q)
$(VFVF)$ (p, q)[4]
$(VVFF)$ (p, q)[5]
$(FFFV)$ (p, q)
$(FFVF)$ (p, q)
$(FVFF)$ (p, q)
$(VFFF)$ (p, q)
$(FFFF)$ (p, q)

$N^{(0)}{}'(\bar{\xi}) = \bar{\xi}$ Def.,
$N^{(n)}{}'(\bar{\xi}) = N'N^{(n-1)}{}'(\bar{\xi})$ Def.

Este esquema muestra con claridad que (I) *todas* "[l]as funciones de verdad pueden ordenarse en series" (5.1), y (II) que la lógica proposicional es decidible, completa y consistente. Así, tenemos que tanto dicha serie como la de los números naturales son objetos formales que instancian el concepto formal de

[4] Tal como indica Wittgenstein en 5.101, este signo proposicional está por p.
[5] En este caso, está por q.

forma general de una serie de formas. Con esto, queda expuesta y analizada la perspectiva anti-descriptivista de Wittgenstein respecto al *Tractatus*. Veamos ahora su posición anti-descriptivista en etapas posteriores.

Pinto (2002, p. 139, nota al pie 4), argumenta que las reglas lingüísticas son, en primer lugar, regularidades empíricas que un intérprete discierne en el comportamiento de un hablante; en segundo lugar son generalizaciones universales; y, en tercer lugar, regularidades normativas. Me centraré en el primer sentido propuesto, puesto que, en mi opinión, puede crear confusión. Creo que en este punto habría que diferenciar dos tipos de enunciados:[6]

(a) Una semana está compuesta de siete días.
(b) 'Una semana está compuesta de siete días' expresa una regla semántica del castellano.

El enunciado (a) expresa una regla que define un sistema conceptual (un juego, en el sentido empleado por Wittgenstein), y en este caso expresa una regla lingüística. Pero un enunciado como (b) es un enunciado empírico, el cual no expresa una regla. Las reglas no son susceptibles de verificación o falsación, mientras que un enunciado como (b) sí lo es, pues hay que atender al comportamiento de los hablantes de una comunidad para averiguar si la siguen o no. Pinto, al decir que las reglas *son* regularidades empíricas, parece estar confundiendo enunciados del tipo (a) con enunciados del tipo (b), aunque no ponga ningún ejemplo concreto para ilustrar su posición. Es evidente que una regla gramatical como (a) expresa una definición, mientras que (b) es un enunciado sobre una conducta.[7]

En cualquier caso, Wittgenstein es muy claro al respecto: las reglas gramaticales no rinden cuentas ante ninguna realidad, sea empírica o platónica, algo que deja muy claro en sus obras "intermedias" y "posteriores". Así, en *Philosophical Investigations* (1958), encontramos que Wittgenstein analiza el papel de las reglas gramaticales para el uso de 'no':

---

[6] Agradezco al profesor Alemán sus inteligentes observaciones en este punto mediante correspondencia personal.

[7] Decir, por ejemplo, que el enunciado (a) no es una regla que se siga en la comunidad X, no es falsar empíricamente la regla; del mismo modo que si se sigue no se ha confirmado empíricamente. La razón es simple, no podemos decir que la hemos confirmado porque sin esa regla 'semana' no tendría aún significado. Luego si no es confirmable empíricamente tampoco es falsable. Por tanto, descubrir que se sigue una regla (enunciado (b)) no es lo mismo que la definición de la regla (enunciado (a)) y por ello no es correcto definir una regla gramatical como regularidad empírica.

> No tiene sentido discutir si estas u otras reglas son las correctas para la palabra 'no' (quiero decir, si son adecuadas a su significado). Pues la palabra no tiene aún ningún significado sin estas reglas; y si cambiamos las reglas, obtiene otro significado (o ninguno) y en tal caso podríamos también cambiar la palabra. (Wittgenstein 1958, p. 351).

Veamos, pues, con más detalle, por qué no tiene sentido plantear esta discusión con respecto a las reglas gramaticales. Para ello, hay que acudir a dos obras fundamentales para entender esta noción, a saber: las *Observaciones filosóficas* y la *Gramática filosófica*. En las *Observaciones filosóficas*, Wittgenstein establece que:

> Yo no llamo a una regla de representación una convención si puede quedar justificada en proposiciones, las cuales describen lo que es representado y muestran que la representación es adecuada. Las convenciones de la gramática no permiten que se les justifique mediante la descripción de lo que es representado. *Toda descripción así presupone ya las reglas de la gramática* (Wittgenstein [1975a] 1997, § 7, el énfasis es mío).

Es evidente que las reglas de uso de la palabra 'no' son convencionales en este sentido que emplea Wittgenstein. Pero aún hay más. En *Philosophical Grammar* Wittgenstein argumenta lo siguiente:

> La gramática no tiene que rendirle cuentas a ninguna realidad. Las reglas gramaticales determinan el significado (lo constituyen) y, de esa manera, no son responsables [no dependen] de ningún significado siendo también, en esa medida, arbitrarias (Wittgenstein [1974] 1992, § 133).

Esto es, no se puede apelar al significado de una palabra para justificar (y determinar) si una regla gramatical es adecuada o correcta para dicho significado pues sin esa regla la palabra no tiene aún significado; y es, precisamente, el hecho de que las reglas sean arbitrarias lo que va en contra de la posibilidad de su justificación (Wittgenstein [1974] 1992, § 133). Es decir, una regla no es gramatical si puede ser justificada "por el hecho de que una representación acorde con ella coincida con la realidad" (Wittgenstein [1974] 1992, §134). Por tanto, un criterio claro y seguro para hablar de regla gramatical es su imposibilidad de justificación mediante una realidad (empírica o platónica) independiente de la regla. Otro ejemplo muy claro con respecto a dicha imposibilidad lo encontramos en *Philosophical Grammar* (Wittgenstein [1974]

1992, § 133), cuando Wittgenstein indica que las reglas gramaticales son arbitrarias, al igual que una unidad de medida, en el sentido de que la elección de una unidad de medida (que no es más que un sistema de reglas del tipo '1m = 1000mm') es independiente de las propiedades físicas del objeto de la medida. Por tanto, la gramática es *a priori* (Wittgenstein [1975a] 1997, § 1), y lo es precisamente porque no hay una realidad independiente que pueda servir para justificar la gramática, pues sin ella, las palabras para los colores, por ejemplo, no tendrían aún significado. Toda descripción y relación entre colores presupone ya la gramática.

Volviendo a la matemática, el propio Wittgenstein ([1975a] 1997, p. 48) indica que expresiones matemáticas como '2+2 = 4' son reglas. Concretamente, son reglas que en enunciados empíricos permiten poner '2+2' en lugar de '4' (cf. Wittgenstein [1975a] 1997, p. 82). No hay duda, por tanto, de que las ecuaciones matemáticas son reglas gramaticales, dado que el intercambio entre expresiones no depende de las propiedades de los objetos que aparecen en enunciados empíricos del tipo 'dos piedras + dos piedras = cuatro piedras'. No se puede acudir a una realidad independiente de la regla, como pueda ser la realidad empírica (obviamente, tampoco sería adecuado acudir a una realidad platónica) para justificar la regla, pues el intercambio entre expresiones presupone ya la regla. Por tanto, ver que '2+2' es intercambiable con '4' no depende de ninguna comparación con los hechos; siendo esto último algo que Wittgenstein ya presentó en el *Tractatus* (véase 6.2321).

A modo de conclusión, la posición de Wittgenstein a lo largo de su obra es eminentemente anti-descriptivista, no sólo anti-platonista, una descripción que se queda corta a la luz de lo aquí expuesto. La noción de objeto lógico-matemático es una noción puramente formal. En su etapa posterior, Wittgenstein dejó claro que las reglas que constituyen tales objetos lógico-matemáticos (como los sistemas formales o procedimientos mecánicos finitos que generan las proposiciones o los números naturales) son reglas gramaticales que no rinden cuentas ante ninguna realidad. Esta perspectiva se puede aplicar al antecedente histórico del *convencionalismo*, el *formalismo*, tal y como veremos a continuación.

III

Como he indicado en la § I. el formalismo es una concepción de la lógica y de la matemática cuyo principal iniciador fue Hilbert.[8] A comienzos del siglo XX, Hilbert propuso el famoso programa finitista por medio del cual pretendía probar la consistencia de la aritmética empleando métodos finitistas; esto es, que no pueda darse el caso de que una proposición *A* y su negación *no A* sean ambas teoremas. Asimismo, el segundo programa de Hilbert estaba relacionado con el llamado 'problema de la decisión' (*Entscheidungsproblem*). Tal problema consistía en establecer un procedimiento de decisión (*Entscheidungsverfahren*) que permitiera determinar o decidir, en un número finito de operaciones, si una expresión o fórmula bien formada es o no válida. Es bien conocido que tales esfuerzos fueron cercenados por los trabajos de Gödel (1931), Church (1936), Turing (1937) y Post (1943).

Estos autores trataron de caracterizar formalmente la noción de procedimiento mecánico finito (esto es, de algoritmo) y resolver así el problema de la decisión, presentando diferentes respuestas consistentes en sistemas formales tales que por medio de un conjunto de reglas y dado un *input*, obtuvieran un *output*, valor o resultado, a través de un conjunto finito de pasos.

Entre estos intentos, podemos destacar, como respuesta directa o indirecta a dicho problema: (a) las funciones recursivas primitivas (Skolem 1923, Gödel 1931)[9], generales (Gödel 1934)[10] y parciales (Kleene 1938, 1943, 1952);[11] Las

---

[8] Wittgenstein ([1974] 1992, [1975a] 1997) también se mostró crítico con respecto a la propuesta de Hilbert denominada 'metamatemática', considerada ésta como una metateoría de la matemática; esto es, primero se formalizaba la teoría matemática que se quería estudiar para después considerar tal teoría el objeto de un estudio matemático. Para Wittgenstein, lo que Hilbert ofrecía era un cálculo como cualquier otro. Es decir, no ofrece un meta-cálculo, sino un cálculo.

[9] Un ejemplo de función recursiva primitiva (adición): $a+0 = a$, [si se parte desde '1', $a+1 = a'$] Def.; $a+(b+1) = (a+b)+1$ Def.

[10] Un ejemplo de función recursiva general: $\varphi(0,y) = \psi(y)$ Def., $\varphi(x+1, 0) = \chi(x)$ Def., $\varphi(x+1, y+1) = \varphi(x, \varphi(x+1, y))$ Def. Esta clase incluye las primitivas, pues estas últimas no son las únicas funciones totales que pueden definirse recursivamente. La diferencia fundamental con respecto a las primitivas es que la recursión se aplica sobre, al menos, dos variables simultáneamente (véase Gödel 1934, p. 69).

[11] Una clase que recibe su nombre del hecho de que una función no necesita estar definida para todas las *n*-tuplas de números naturales que toma como argumentos y que incluyen a las funciones recursivas primitivas y a las funciones recursivas generales como aquellas funciones parciales definidas para todas las *n*-tuplas de números naturales que toma como argumentos. Tal clase es extensionalmente equivalente al formalismo de Turing, lo que me permitió definir la Tesis de *Turing-Kleene* ("una función es computable si y sólo

diferentes clases de funciones recursivas trataron de capturar la clase de las funciones efectivamente computables, para las que existía un procedimiento efectivo de cálculo.[12] En este sentido, la recursión mostró ser un método por medio del cual las funciones podían ser definidas y, a partir de ahí, efectivamente calculadas.

(b) Turing también formalizó la noción de computación mediante el formalismo conocido como la máquina de Turing (1937). En su caso, la Tesis de Turing, asevera que una función es computable si y sólo si es computable por una máquina de Turing (Soare 2009, p. 373). Turing (1937) presentó un sistema formal que consistía en una máquina de carácter abstracto equipada con cinta potencialmente infinita dividida en cuadrados, cada uno de los cuales podía expresar un símbolo. Asimismo, la máquina cuenta con una cabeza lectora que escanea un único cuadrado de la cinta cada vez, con una serie finita de condiciones ($q_1 ... q_n$), que son las distintas configuraciones de la máquina; una serie de símbolos (por ejemplo, 0 y 1); y un conjunto de operaciones (como por ejemplo, escribir un nuevo símbolo; borrar un símbolo escrito; o mover su cabeza lectora un cuadrado hacia la izquierda o hacia la derecha).[13]

---

si es computable por una función recursiva parcial o, de forma equivalente, por una máquina de Turing" y la *Tesis de Kleene*, incluida en la anterior (Mota 2013, §2). La primera muy bien podría sustituir a la *Tesis de Church-Turing*, dados los errores cometidos por Church (Soare, 2009).

[12] No confundir el hecho de que se formularan diferentes CLASES de funciones recursivas con DISTINTOS significados de recursión, como parece hacer Tomalin (2011), que propone tantos significados de recursión como clases hay –ignorando la clase de las funciones recursivas parciales– lo cual es un error. El significado de 'recursión' es el mismo, sólo cambia el alcance de la clase; todas ellas se definen por medio de los esquemas I-V (junto con el esquema VI en el caso de las clases de las generales y las parciales) y, por tanto, haciendo uso de la definición por inducción o por recursión (esquema V) (cf. Kleene 1943, 1952). Nótese, por otro lado, que la inducción también puede ser expresada por medio de la iteración. Así, siguiendo a Wittgenstein ([1974] 1992, § 32), éste nos dice que es claro que debe haber una "prueba" iterativa que "transmita la idea de que así debe ocurrir con todos los números"; esto es, una regla que concuerda con la inducción: $f(1) = a+b$; $f(1) \times (a+b) = (a+b)^2 = f(2)$; $(a+b)^n = f(n)$; $f(n) \times (a+b) = f(n+1)$. Así "una vez que se tiene la inducción, todo ha terminado" (Wittgenstein [1974] 1992, p. 32).

[13] Así, es frecuente expresar las instrucciones de la máquina como sigue: [E_0, 0, E_1, 1]. Esta cuádrupla expresa que la máquina está en el estado E_0, escaneando el símbolo '0' y pasa al estado E_1 al sustituir '0' por '1'. Esta manera de proceder es iterativa dado que se llevan a cabo diferentes operaciones de manera sucesiva, pasando de una configuración a otra (esto es, de un estado a otro), pero es posible definir recursivamente cada término generado. Además, aunque el formalismo de Turing es considerado como un modelo de computabilidad, esto no niega que se pueda concebir como una adecuada formulación de un sistema formal o, también, de un procedimiento mecánico finito (véase Gödel 1964, pp. 71-72).

(c) Por su parte, Post (1921, 1943, 1944) definió un formalismo que consiste en un sistema capaz de generar todas las proposiciones de la lógica de enunciados, y, de ahí, todas las tautologías. Así, sea *g* un enunciado de la lógica proposicional y *P* una variable operacional aplicada a dicho enunciado, el sistema produce un enunciado *g'* que sustituye a *g*. Esto se expresa en su sistema de producción normal como sigue: '$gP \rightarrow Pg$' (Post 1943, p. 199).[14] Este procedimiento de decisión permite decidir (al igual que el de Wittgenstein) en un número finito de pasos si una proposición es o no una verdad lógica (alias "proposición de la lógica" o "tautología").

Wittgenstein, en su *Philosophical Remarks* ([1975a] 1997, § 154), hace una observación de gran interés y pertinencia en este punto. Un sistema formal, como lo son todos y cada uno de los sistemas que he mencionado, es una serie de formas (esto es, una serie ordenada por relaciones internas (4.1252)).[15] Si son series de formas, entonces caen bajo la forma general de una serie de formas, que en este caso, podemos formular como la forma general de un procedimiento mecánico finito (lo cual no expresa sino una variable): $[\tilde{\sigma}, \tilde{\varepsilon}, \tilde{O}(\tilde{\varepsilon})]$ (Mota 2013, 2014). Esta expresión recoge lo que es esencial a todo procedimiento mecánico efectivo, esto es, lo que todos ellos tienen en común.[16] En relación con los que he mencionado, $\tilde{\sigma}$ estaría por un conjunto de ítems iniciales, tales como números naturales, proposiciones elementales, axiomas; $\tilde{\varepsilon}$ sería un subconjunto de tales ítems iniciales y $\tilde{O}(\tilde{\varepsilon})$ sería un término generado a partir de un elemento previamente computado $\tilde{\varepsilon}$ al que se le aplica la variable operacional $\tilde{O}(\ )$, cuyos valores pueden ser las operaciones de la aritmética, las operaciones lógicas o las operaciones que realiza una máquina de Turing, por poner unos ejemplos. Así, para todos ellos es posible instanciar una serie como '$\tilde{\varepsilon}$, $\tilde{O}\tilde{\varepsilon}$, $\tilde{O}\tilde{O}\tilde{\varepsilon}$...' (cf. Mota 2013, 2014):

---

[14] Otras formulaciones también fueron relevantes: véase Marion (1998), Odifreddi (2001) y Frascolla (1994) para diferentes análisis de las relaciones entre las propuestas de Church (1932, 1936) y su cálculo-λ, por un lado, y las propuestas de Wittgenstein en el *Tractatus*, por otro. También recomiendo Rodych (1999, 2002, 2003), Wrigley (1977) y Shanker (1987), que muestran el conocimiento que Wittgenstein tenía sobre los avances en la Teoría de la computabilidad.

[15] Por otro lado, como Wittgenstein (1922) indica en 5.232, "[l]a relación interna que ordena una serie equivale a la operación mediante la que un término resulta a partir de otro".

[16] De manera análoga a la forma general de una proposición, que recoge lo que es esencial a toda proposición, lo que todas ellas tienen en común (1922, 5.471).

(a) En el caso de una función recursiva

$\tilde{\varepsilon}$, $\tilde{O}\tilde{\varepsilon}$, $\tilde{O}\tilde{O}\tilde{\varepsilon}$,...

(2+2), (2+2)+1, ((2+2)+1)+1,...

(2+2), (2+3), (2+4),...

Tenemos pues una serie de definiciones del tipo

2+4 = (2+3)+1.

(b) En el caso del formalismo de Post (muy parecido al de Wittgenstein)

$\tilde{\varepsilon}$, $\tilde{O}\tilde{\varepsilon}$, $\tilde{O}\tilde{O}\tilde{\varepsilon}$,...

g, *Pg*, *PPg*,...

que en el caso de Wittgenstein sería

$\tilde{\xi}$, *N*'($\tilde{\xi}$), *N*'*N*($\tilde{\xi}$),...

Esta serie de definiciones la podemos formular como sigue (sólo muestro el paso recursivo):

$P^{(n)}$'(*g*) = *P*'$P^{(n-1)}$'(*g*)./ $N^{(n)}$'($\tilde{\xi}$) = *N*'$N^{(n-1)}$'($\tilde{\xi}$).

(c) En el caso del formalismo de Turing

$\tilde{\varepsilon}$, $\tilde{O}\tilde{\varepsilon}$, $\tilde{O}\tilde{O}\tilde{\varepsilon}$,...

E_0, $\tilde{O}(E_0)$, $\tilde{O}\tilde{O}(E_0)$,...

que en el caso de la generación de los números naturales se puede escribir:

E_0; Sub0/1' Esc0' (E_0); >' Esc1' (Sub0/1' Esc0' (E_0)),[17]... que significa:
E_0, E_1 (=Sub0/1' Esc0' (E_0)), E_2 (= >' Esc1' (Sub0/1' Esc0' (E_0))),... y así sucesivamente.[18]

Por tanto, cada estado/configuración o término se puede definir como sigue:

$\tilde{O}^{(n)}$'($\tilde{\varepsilon}$) = $\tilde{O}$'$\tilde{O}^{(n-1)}$'($\tilde{\varepsilon}$).

[17] 'EscX' significa 'escanear X'; 'SubX/Y' significa 'sustituir X por Y'; '>' significa 'mover a la derecha'.

[18] Por ejemplo, E_1 es el estado de la máquina obtenido estando previamente en E_0, escaneando 0, y substituyendo 0 por 1.

Así, la expresión 'Sub0/1' Esc0'>' Esc1' (Sub0/1' Esc0' (E_0))' muestra que, estando en E_0 (=0), se aplican las operaciones 'Esc0' y 'Sub0/1', lo que permite calcular '1', pasando así al estado E_1. Mediante la aplicación de las operaciones 'Esc1', '>', 'Esc0' y 'Sub0/1' la máquina calcula '2' y así sucesivamente. La recursión caracteriza la definición de los *n* estados/configuraciones, términos o formas (v.gr., '$\tilde{O}^{(n)}$'($\tilde{\varepsilon}$)') efectivamente computados por una máquina de Turing. Esto caracteriza *qué* hace una máquina de Turing (lo que se puede aplicar al resto de formalismos), esto es, generar recursivamente una serie de términos, pero *cómo* lo hace queda descrito por medio de la iteración, esto es, por medio de la repetición sucesiva de operaciones sobre el último resultado calculado (cf. Mota 2013, 2014). Por tanto, para todos ellos es posible formular la forma general de un procedimiento mecánico finito por medio del siguiente sistema de reglas (Mota 2013, 2014):

$\tilde{O}^{(0)}$'($\tilde{\varepsilon}$) = $\tilde{\varepsilon}$ Def.,
$\tilde{O}^{(n)}$'($\tilde{\varepsilon}$) = $\tilde{O}$'$\tilde{O}^{(n-1)}$'($\tilde{\varepsilon}$) Def.

En Mota (2013, 2014) mostré que la forma general, que es un concepto formal, muestra lo que todos ellos tienen en común. Por tanto, todos estos procedimientos mecánicos efectivos de decisión son instancias, objetos matemáticos, que caen bajo dicha forma general.[19] Tales objetos matemáticos son sistemas formales, cálculos, construidos y constituidos por reglas gramaticales.[20]

---

[19] Las diferencias intensionales (esto es, cómo proceden u operan) son importantes para ver, por ejemplo, qué formulación puede ser más clara o perspicua, pero no es esencial a la formulación de la forma general de un procedimiento mecánico finito, entendido como concepto formal. Así, no es esencial que se implemente recursivamente (como las funciones recursivas) o iterativamente (como la máquina de Turing; siendo posible tanto definir cómo implementar iterativamente una función definida por recursión; véase Robinson 1947, Abelson *et al*. 1996). Del mismo modo que para la formulación de la forma general de una proposición no son esenciales las diferencias intensionales entre proposiciones. La razón es que la forma general expresa lo que todos ellos tienen en común (Mota 2013, 2014). Por otro lado, el 'último' Wittgenstein enfatiza que los procedimientos muestran los llamados 'parecidos de familia', una propuesta más anti-esencialista que la de la primera etapa. En todo caso, me parece muy relevante y acertada la apreciación de que los procedimientos mecánicos finitos son después de todo, series de formas.

[20] Una definición recursiva es una regla gramatical que, como Wittgenstein ([1975a] 1997, §163) señala en *Philosophical Remarks*, es una regla fundamental del sistema que nos indica cómo puedo proceder (y como tal, no se puede aseverar o negar). Del mismo modo, en *Philosophical Grammar* ([1974] 1992), nos indica que "es una regla para la

Por tanto, desde la perspectiva de Wittgenstein, tenemos que deshacer la imagen de la matemática como ciencia cuyas expresiones describen (y descubren) objetos matemáticos y hechos acerca de éstos, como puedan ser, series o procedimientos mecánicos finitos. Todos estos objetos matemáticos formales, o sistemas formales, son invenciones o construcciones matemáticas, constituidos por reglas gramaticales. Las ecuaciones matemáticas no son proposiciones susceptibles de verificación o falsación en contraste con una realidad empírica o platónica, como defienden los *descriptivistas.* Son reglas gramaticales que fijan el significado de ciertos signos (no aseveran o niegan nada, no son una figura de ninguna realidad). Estar en desacuerdo con una regla como '2+2 = 4' (la cual muestra que podemos substituir '2+2' por '4' constituyendo, de este modo, el significado de ambas expresiones) no es negar una regularidad que está de acuerdo con los hechos; es o bien no comprender los significados de '2+2' y de '4', o bien atribuir otro significado (o ninguno) a tales expresiones.

IV

En éste último apartado, voy a comentar brevemente algunas cuestiones que se desprenden del análisis presentado en las secciones anteriores y a explicar cómo se aplican a propuestas que, haciendo uso de cálculos, conciben los dominios cognitivos como órganos mentales (Chomsky 2007a). Centraré mi análisis en el programa chomskiano. En concreto, analizaré el procedimiento generativo/computacional en términos de un formalismo y la noción de representación mental como un concepto formal, interno a la notación conceptual del sistema, del cálculo, y defenderé que no comporta ninguna carga ontológica. En este sentido, no son representaciones *de* nada.

Desde sus primeros escritos, Chomsky (1959, p. 143) ha considerado que, desde un punto de vista conceptual, la teoría de la gramática puede verse como el estudio de una clase especial de funciones recursivas. Esta idea persiste de una u otra manera en diferentes trabajos, como por ejemplo, en *The Logical Structure of Linguistic Theory* (Chomsky 1975), en el que reconoce la importancia de los avances de la Teoría de la Computabilidad para el estudio del lenguaje

construcción de reglas de sustitución, o también el término general de una serie de definiciones [formas]" (36, p. 851).

natural, o, más recientemente, en Chomsky (2007b), donde indica que uno de los principales factores que han impulsado el desarrollo de la perspectiva biolingüística ha sido el trabajo realizado dentro de la Teoría de la Computabilidad, que ha permitido estudiar más seriamente los mecanismos formales de la gramática generativa. En esta misma línea, Chomsky (2012) ha indicado que el estudio del lenguaje-I, entendido como un sistema dotado de infinitud discreta, cae dentro de la Teoría de la Computabilidad, siendo, de este modo, visto como un procedimiento computacional que genera recursivamente una cantidad potencialmente infinita de expresiones jerárquicamente estructuradas, interpretadas por los sistemas sensorio-motor y conceptual-intencional (Chomsky 1995a p. 226). Tal procedimiento está basado en *Merge*.

La forma general de *Merge* se puede formular de la siguiente manera, siguiendo la notación de Wittgenstein: [*N*, O_s, M(O_s)]. Así, *N* hace referencia a ítems léxicos y objetos sintácticos que configuran la numeración, O_s hace referencia a un objeto sintáctico cualquiera de la serie generada y M(O_s) hace referencia a un objeto sintáctico nuevo generado a partir de O_s mediante la aplicación de M ( ).[21] Así, y de manera esquemática, vemos que lo que *Merge* hace (esto es, generar/definir recursivamente objetos sintácticos, independientemente de su estructura interna, como vemos en la definición formal) queda expresado mediante la serie 'O_s, M'(O_s), M'M'(O_s),...'.[22] Esto puede definirse recursivamente de manera formal como sigue (Mota 2013, nota 2, 2014):

$$M^{(0)}\text{'}(O_s) = O_s,$$
$$M^{(n)}\text{'}(O_s) = M\text{'}M^{(n-1)}\text{'}(O_s).$$

Por otra parte, Chomsky (2007a, p. 6, 2008, p. 139) ha insistido en que el modo de operar (o de implementarse de manera abstracta) de este procedimiento generativo (esto es, cómo procede) es aplicando iterativamente la operación

[21] Sea la numeración *N* = {*Juan, canta, muchas, canciones*}, lo que *Merge* hace es tomar dos ítems léxicos (definida aquí como una operación binaria) 'muchas' y 'canciones' y forma el objeto sintáctico {*muchas*, *canciones*} = {X,Y}. Mediante otra aplicación, *Merge* genera un nuevo objeto sintáctico {*canta*, {*muchas*, *canciones*}}, formado a partir de un objeto sintáctico previamente computado ({*muchas*, *canciones*}); por último, *Merge* genera {*Juan*, {*canta*, {*muchas*, *canciones*}}}.

[22] En el ejemplo: $O_{s\{muchas,\ canciones\}}$; $M_{canta}\text{'}O_{s\{muchas,\ canciones\}}$;$M_{Juan}\text{'}M_{canta}\text{'}O_{s\{muchas,\ canciones\}}$) (Mota 2013, 2014).

Merge. Así, la aplicación iterativa queda descrita mediante la siguiente serie, que muestra, en cada paso, las *n* veces que *Merge* se aplica de forma sucesiva, lo cual puede generar todo tipo de expresiones (con auto-inclusión o no) del lenguaje natural: $M^{(0)}$, $M^{(1)}$, $M^{(2)}$,..., $M^{(n)}$,..., $M^{(n+1)}$; esto es: $M'M^{(n)}'(O_s) = M^{(n+1)}'(O_s)$.

Atendiendo a la definición formal, salta a la vista que el procedimiento mecánico finito propuesto por Chomsky es una instancia de la forma general de una serie de formas, pues una serie de objetos sintácticos es una serie ordenada por relaciones internas. De este modo cae bajo la forma general arriba presentada.[23]

Pero, además de su carácter abstracto, Chomsky (2007a, p. 8, 2010, p. 53), establece que *Merge* es una propiedad del lenguaje genéticamente determinada (2007a p. 8, 2010, p. 53). Así, Chomsky va más allá de la Teoría de la Computabilidad al hablar de la evolución del lenguaje y especular con que el cerebro sufrió una reorganización, quizá por algún evento genético, y ello propició la aparición de *Merge* (Chomsky 2006, p. 184, 2010, pp. 58-59). En este sentido, y partiendo de la idea de que la facultad del lenguaje es un sistema biológico, Chomsky (2005) formula tres factores que se deben tener en cuenta en el desarrollo de la facultad del lenguaje: un primer factor que correspondería con la dotación genética (la Gramática Universal), un segundo factor que hace referencia a la experiencia y un tercer factor que incluye principios no específicos de la facultad del lenguaje.

Dentro del tercer factor, se incluye, entre otros, un subtipo que hace referencia a la eficiencia computacional. Un ejemplo de eficiencia computacional sería la condición de la no-injerencia (*no tampering*), esto es, tomando X e Y, *Merge* genera {X,Y} = Z, dejando X e Y sin modificaciones, sin añadir o quitar elementos (Chomsky 2010, p. 52). Sin embargo, las condiciones de eficiencia computacional sólo definen el cálculo, el sistema formal. Chomsky, en mi opinión, parece mezclar distintos niveles de análisis. Sin duda, la

[23] En una correspondencia personal con Chomsky en la que le planteé esta cuestión, respondió que no dudaba de que *Merge* podía formularse fácilmente dentro de un concepto general de procedimiento mecánico finito. Expuso que, entendiendo *Merge* como una operación binaria, tenemos dos posibilidades lógicas: *Merge* Externo (X distinto de Y) y *Merge* Interno (X es un término de Y). Esto es absolutamente válido. Sin embargo, la forma general de *Merge* arriba mostrada ya incluye tales posibilidades y no es necesario expresar tales posibilidades de manera explícita, pues M(Os) tiene la misma forma general Os =X,Y, tanto si X e Y son distintos como si X es un término de Y.

capacidad/facultad del lenguaje puede entenderse como un objeto natural, susceptible de investigación empírica, pero no así el procedimiento mecánico finito construido para su caracterización abstracta, pues éste no es más que un cálculo, susceptible de una investigación formal (matemática). Las propiedades del cálculo (como la recursión) están determinadas *a priori* por las reglas que lo constituyen (cf. Mota 2013, 2014).

Anthony Kenny (1989, 1990) expuso esta cuestión de una manera clara, al proponer que la mente es "la capacidad de adquirir habilidades intelectuales" (Kenny 1990, p. 204). En este sentido, hay que distinguir entre poseedores y vehículos. El poseedor de la habilidad lingüística es un ser humano y el vehículo "es la parte de su poseedor por virtud de la cual éste es capaz de ejercer la habilidad...algo concreto y más o menos tangible" (Kenny 1990, p. 205). Podemos hacer una distinción entre las personas (el poseedor), la mente (conjunto de capacidades) y el cerebro (el vehículo de tales capacidades). Evidentemente, las personas y sus cerebros son objetos físicos, pero no así la mente, que es un conjunto de capacidades. Conviene tener presente también que tal distinción no es una distinción metafísica/ontológica, sino conceptual, en virtud de la cual se definen 'mente' y 'vehículo'. En este sentido, Kenny propone que pensemos en una calculadora como analogía a lo que se acaba de exponer: el fisiólogo estaría a la par junto con el ingeniero electrónico, mientras que el psicólogo está en una posición análoga a la del matemático, pues intenta descubrir el algoritmo que usa la calculadora (1990, p. 206). La calculadora tiene la habilidad de computar un algoritmo y su análisis exige una investigación matemática, más que electrónica. El aparataje electrónico sería el vehículo.

Sin embargo, si atendemos al *naturalismo metodológico* que defiende Chomsky, el lenguaje, entendido como un órgano *mental* o un sistema biológico, ha de ser considerado como un objeto real, esto es, como el resto de objetos del mundo, y su estudio ha de ser abordado desde una perspectiva naturalista, esto es, como las ciencias naturales estudian otros objetos del mundo. Parecería, por tanto, que esto resulta incompatible con el anti-realismo que vengo exponiendo a lo largo de este artículo. Profundicemos más en esta cuestión. Bajo el naturalismo metodológico, tal y como acabo de exponer, Chomsky hace una aproximación a la mente bajo la cual considera el lenguaje como un elemento del mundo natural (Chomsky 1995b, p. 1). Chomsky subraya aquí que usa el término 'mente' y

'mental' sin ninguna implicación metafísica, estando a la par de términos como 'químico', 'óptico' o 'eléctrico' (Chomsky 1995b, p. 1). Tales términos seleccionan ciertos aspectos del mundo como foco de investigación, entendiendo por 'mente' los aspectos mentales del mundo (Chomsky 1995b, p. 1). Desde la perspectiva biolingüística, nos dice Chomsky (2006), el lenguaje es un componente de la mente, entendiendo 'mente' en el sentido de los científicos del siglo XVIII, de tal manera que, no siendo posible postular un problema mente/cuerpo coherente, sólo podemos considerar los aspectos del mundo denominados mentales como el resultado de una organización orgánica tal como la del cerebro (Chomsky 2006, p. 173, véase también 2005, p. 2).

Estas consideraciones generales han sido algo más precisadas en un interesante trabajo reciente. En dicho trabajo, Berwick, Friederici, Chomsky & Bolhuis (2013, p. 90), nos indican que el lenguaje humano está asentado sobre un mecanismo computacional particular realizado neurológicamente. En este primer apunte, es interesante notar que el mecanismo computacional no sólo es una herramienta construida para caracterizar el lenguaje de manera abstracta, sino que se postula que tal procedimiento mecánico finito está realizado neurológicamente. Es decir, tal procedimiento está a la par que, por ejemplo, las neuronas; son objetos del mundo. Por decirlo de otra manera, dicho procedimiento no sólo es una clase especial de funciones recursivas (ver supra), sino que también es un objeto del mundo.

Como señalé unas líneas más arriba, el procedimiento mecánico finito basado en *Merge* genera objetos sintácticos (de los cuales me ocuparé más abajo), que son interpretados por dos sistemas, el sensorio-motor (encargado entre otras cosas de la externalización) y el intencional-conceptual (que hace referencia, entre otras cuestiones, al pensamiento y a la planificación de la acción) (Chomsky 2010, p. 269). Obviamente, en el plano neurológico, tal mecanismo generativo/computacional debe diferenciarse de ambos sistemas de interfaz. Berwick *et al.* (2013, p. 93) presentan diferentes regiones identificadas tanto con el mecanismo computacional como con los sistemas de interfaz. Así, dicho mecanismo está sustentado (o realizado neurológicamente) por el área de Brodmann 44 y el córtex temporal superior posterior. El sistema sensorio-motor, por su parte, estaría sustentado por el córtex premotor y el córtex temporal superior. Finalmente, el procesamiento semántico (propio del sistema

conceptual-intencional) está sustentado por el área de Brodmann 45 y por el córtex frontal inferior, así como por porciones del córtex temporal. Así, nos dicen Berwick *et al.* (2013, p. 93), el estudio de las bases neurológicas del lenguaje debe considerar aquellas partes o regiones particulares del cerebro que representan el mecanismo computacional así como aquellos sistemas de interfaz. Por tanto, cada uno de estos sistemas consisten en las mencionadas regiones particulares del cerebro, conectadas vía tractos específicos (véase Berwick *et al.* 2013 para más detalles).

Sin embargo, creo que tal estudio, aunque muy interesante y desde luego legítimo en relación con la facultad del lenguaje, no es un obstáculo para el anti-realismo que vengo exponiendo. El estudio del lenguaje en este nivel neurológico es, siguiendo a Kenny, el estudio del vehículo, del cerebro. El mecanismo computacional (cuyo estudio apropiado es una investigación matemática) no consiste en estas o aquellas regiones cerebrales. Tal procedimiento es una construcción matemática que no supone el descubrimiento de ningún objeto mental del mundo natural. El estudio del cerebro, por su parte, sí es el estudio de un objeto del mundo natural, susceptible de una metodología naturalista. Eso sí, el estudio se realiza en otro nivel de análisis. De manera análoga, Alemán (2011) indica, en su análisis introductorio a una primera versión del argumento de indispensabilidad, que:

> [P]uesto que una ciencia como la física describe y predice con éxito lo que acontece en la realidad circundante, tal teoría tiene que ser verdadera y si esto es así, entonces tienen que existir las entidades a las que se refieren los términos de la teoría (Alemán 2011, p. 23).

Así, continúa Alemán (2011, p. 23), no sólo tienen que existir los electrones y los *quarks*, sino que también tienen que existir los números, funciones, [...], referidos por los términos matemáticos empleados en la formulación de tal teoría física. Así, una cosa es que construyamos un sistema formal, un cálculo, y llevemos a cabo un análisis matemático del mismo, y otra cosa completamente diferente es que (I) lo postulemos como un objeto de una realidad (empírica o platónica), y (II) entendamos que ambos niveles de análisis, a saber, el lógico-matemático y el empírico en este caso, no sólo están al mismo nivel, sino que son el mismo; esto es, las entidades físicas y matemáticas existen en la misma

realidad espacio-temporal y las abordamos mediante una metodología naturalista. Esta concepción realista del mecanismo formal propuesto por Chomsky parece estar detrás de la idea de que dicho mecanismo es un objeto del mundo natural.

Sin embargo, una cosa es decir que el lenguaje es un sistema biológico y otra muy diferente decir que la caracterización abstracta hecha mediante la formalización planteada por Chomsky sea un objeto real del mundo natural; es decir, que tal formalización describa un objeto del mundo natural. Tal formalismo es, en cierto sentido, autónomo con respecto a cualquier realidad en el sentido de que no es descriptivo de ella. En este sentido, en el momento en el que se construye un formalismo se están constituyendo tanto los signos lógico-matemáticos empleados en éste como las reglas para su uso dentro del sistema (por ejemplo, un sistema formal o procedimiento mecánico finito, puede calcular en un número finito de pasos, definidos unos axiomas, otras verdades lógicas mediante la aplicación de operaciones siguiendo un conjunto de reglas, pudiendo ordenarse tales verdades en series; cf. supra). De este modo, no se descubren los objetos mentales como el mecanismo generativo/computacional propuesto por Chomsky, sino que se crean, se construyen y se constituyen mediante las reglas gramaticales.

Relacionado con esto último, conviene recordar que una regla gramatical no niega ni asevera nada y por tanto no describe ninguna realidad. Así habría que distinguir entre (I) la expresión "2+2" se puede intercambiar con la expresión "(2+1)+1", (II) el sujeto S usa la regla 'R' y (III) el sujeto S usa la regla 'R''. Así, (I) no es un enunciado empírico y por tanto, las propiedades constituidas por dicha regla gramatical no dependen de ninguna realidad; son autónomas frente a ésta y se constituyen por la regla y, por tanto, dentro del sistema. Por su parte, los enunciados (II) y (III) son enunciados empíricos y, por tanto, lo que se confirma o desconfirma son tales enunciados, no las reglas 'R' o 'R''. Dichas reglas, así como (I), no dicen nada, no describen ninguna realidad.

A continuación, y para terminar, veremos en qué medida podemos decir que las representaciones mentales son objetos del mundo natural o, por el contrario, constituyen *objetos* mentales no descubiertos, sino creados mediante la formulación, creación o invención de un formalismo empleado para caracterizar la facultad del lenguaje. Con respecto a la noción de representación mental,

Chomsky (1980, p. 5) indica que tal noción caracteriza de manera abstracta las propiedades de ciertos mecanismos físicos, pero que no conllevan ningún importe ontológico; esto es, no es necesario postular la existencia de entidades que pertenezcan a una realidad diferente a la del mundo físico.[24] En *Language and Nature* (Chomsky 1995b) ofrece una buena definición de lo que él entiende por representación mental. Así, Chomsky (1995b, p. 52), dice que no hay ninguna cuestión con sentido que tenga que ver con el contenido de una representación interna; ninguna noción como 'contenido' o 'representación de' es considerada dentro de la teoría y, por ello, no tiene sentido plantearse la pregunta sobre su naturaleza. En el caso del estudio del lenguaje, también se habla de representaciones de varios tipos, por ejemplo, representaciones semánticas y fonéticas que son enviadas a otros sistemas como son el sensorio-motor y el conceptual-intencional. Pero tampoco en este caso, insiste Chomsky (1995b, p. 53), tenemos que buscar alguna construcción objetiva de las que sean "representaciones de".

Más recientemente, Berwick *et al.* (2013, p. 91), recuerdan que el mecanismo computacional formulado para el lenguaje humano incluye una operación, *Merge*, que construye nuevos elementos representacionales *Z* desde otros elementos previamente construidos, a saber, *X*, *Y*. Continúan indicando que una representación interna tal presenta, como señalé más arriba, una versión ordenada para ser externalizada. Por su parte, las computaciones que construyen tanto las representaciones sintácticas por el mecanismo generativo/computacional, como su versión conceptual-intencional (semántica) aluden a lo que Berwick *et al.* (2013, p. 89), denominan internalización. Relacionado con esto último, Berwick *et al.* (2013, pp. 93-94) indican que se pueden señalar determinadas regiones cerebrales a modo de correlatos neuronales que sustentan tal construcción de representaciones sintácticas (cf. Berwick *et al.* 2013, pp. 91-92 para más detalles, por ejemplo, sobre el algoritmo que etiqueta los elementos representacionales). En este sentido, el área de Broca, en concreto, el área de Brodmann 44 (ver supra), se activa durante el

[24] Autores como Fodor (1992) defienden el realismo de las representaciones mentales, entendidas como actitudes proposicionales, indicando que uno es realista en cuanto a las actitudes proposicionales si (a) sostiene que hay estados mentales cuya ocurrencia e interacción causan el comportamiento y (b) sostienen que tales estados mentales son semánticamente evaluables.

procesamiento de estructuras jerárquicas en el lenguaje natural (Berwick *et al.* 2013, p.94), que también activa el córtex temporal superior posterior, conectados por medio del fascículo arqueado, y partes del fascículo longitudinal superior (Berwick *et al.* 2013). En todo caso, tal sistema dorsal (que incluye el área Broca, esto es, el área de Brodmann 44) sustenta las computaciones basadas en reglas para la construcción de estructuras jerárquicas. Esto implica cierto realismo sobre las representaciones mentales, tal como Pylyshyn (1991) indicó en un interesante trabajo, ya clásico. Así, las representaciones mentales, como puedan ser las representaciones mentales sintácticas o conceptuales-intencionales, han de estar realizadas neuronalmente y, por ello, no es necesario, como indica Chomsky, postular la existencia de entidades que pertenezcan a una realidad diferente a la del mundo físico. Según esta perspectiva, éstas están, por tanto, explícitamente codificas en el sistema físico, esto es, el cerebro.

Sin embargo, la crítica a este realismo puede hacerse siguiendo la misma crítica presentada unas líneas más arriba, a saber, una cosa es que las construcciones lingüísticas tengan una realización neurológica y otra muy diferente es que las denominadas representaciones mentales tengan el mismo estatus ontológico que el de las neuronas. Una vez más, al construir un formalismo que mediante un conjunto de reglas construye objetos sintácticos, se está, de hecho, creando, no descubriendo, tanto las reglas gramaticales como los objetos sintácticos que, siguiéndolas, se constituyen en dicho cálculo. Los objetos sintácticos o representaciones mentales (v.gr. {X,Y} o $M'(O_s)$) son, por tanto, objetos construidos en el momento en el que creamos un formalismo que caracteriza de manera abstracta la facultad del lenguaje. Su aplicación es interna al sistema. De esto se deprende que *lo mental* no es un mundo independiente del mundo natural, pero tampoco es una parte de éste en el sentido de que es una porción que ha de ser descubierta; es, más bien, un sistema formal creado para caracterizar de manera abstracta determinadas capacidades. Dicha construcción implica la creación de otros "objetos", cuyas propiedades y aplicaciones se constituyen dentro del sistema.

En suma, el aspecto fundamental desde la perspectiva wittgensteiniana que vengo defendiendo es que si concebimos *Merge* como un procedimiento mecánico finito, entonces hay que entenderlo como una construcción

matemática, no como algo determinado genéticamente. Por otro lado, los inputs que *Merge* toma como argumentos y los valores que, al aplicar una operación a través de una serie de pasos finitos, devuelve, pueden entenderse como representaciones mentales/objetos sintácticos. Pero esto no debe hacer creer que estos objetos existen en una realidad platónica, punto en el que estoy de acuerdo con Chomsky, pero tampoco en una realidad física o empírica, como Chomsky parece sugerir. No son objetos del mundo natural descubiertos, únicamente son construcciones interna al formalismo creado para caracterizar de manera abstracta una capacidad intelectual.

Conclusiones

En este trabajo he pretendido mostrar que adoptando la perspectiva anti-realista de Wittgenstein, que defendió tanto en el *Tractatus* como en sus obras intermedias y posteriores, podemos tratar los objetos/sistemas matemáticos formales como invenciones, creaciones o construcciones (Wittgenstein 1978) y no como un sistema de proposiciones que describen una realidad empírica (realismo empírico) o platónica (realismo platónico), sino como un sistema de reglas gramaticales que constituyen el cálculo y el significado (el uso) de los signos y símbolos en él implicados. Un ejemplo de la utilidad de la formulación wittgensteiniana la he mostrado en su aplicación al procedimiento mecánico finito definido por Chomsky, así como a otras nociones (como la de representación mental) implicadas en el mismo.

Bibliografía

Alemán, A. (2011), *Lógica, matemáticas y realidad*, Madrid: Tecnos.

Berwick, R., Friederici, A.D., Chomsky, N. y J. Bolhuis (2013), "Evolution, Brain, and the Nature of Language", *Trends in Cognitive Science* 17: 89-98.

Chomsky, N. (1959), "On Certain Formal Properties of Grammars", *Information and Control* 2: 137-167.

Chomsky, N. (1975), *The Logical Structure of Syntactic Theory*, Nueva York: Plenum Press.

Chomsky, N. (1980), *Rules and Representations*, Nueva York: Columbia University Press.

Chomsky, N. (1995a), *The Minimalist Program*, Cambridge, MA: MIT Press.

Chomsky, N. (1995b), "Language and Nature", *Mind* 104: 1-61.

Chomsky, N. (2005), "Three Factors in Language Design", *Linguistic Inquiry* 36: 1-22.

Chomsky, N. (2006); *Language and Mind*, Cambridge: Cambridge University Press.

Chomsky, N. (2007a), "Approaching UG from Below", en Sauerland, U. y H.M. Gärtner (eds.), *Interfaces + Recursion = Language?*, Berlín: Mouton, pp. 1-30.

Chomsky, N. (2007b), "Of Minds and Language", *Biolinguistics* 1: 9-27.

Chomsky, N. (2008), "On Phases", en Freidin, R., Otero, C. y M.L. Zubizarreta (eds.), *Foundational Issues in Linguistic Theory*, Cambridge, MA: MIT Press, pp. 133-166.

Chomsky, N. (2010), "Some Simple Evo Devo Theses: How True Might They Be for Language?", en Larson, R., Déprez, V. y H. Yamakido (eds.), *The Evolution of Human Language*, Cambridge: Cambridge University Press, pp. 45-62.

Chomsky, N. (2012), "Some Core Contested Concepts", *Proceedings of the CUNY* 2012:1-18.

Church, A. (1932), "A Set of Postulates for the Foundation of Logic", *The Annals of Mathematics* 33:346-366.

Church, A. (1936), "An Unsolvable Problem of Elementary Number Theory", en Davis, M. (ed.), *The Undecidable*, Nueva York: Raven Press, pp. 88-107.

Cutland, N. (1980), *Computability: An Introduction to Recursive Function Theory*, Cambridge: Cambridge University Press.

Fodor, J. (1992), *A Theory of Content and Other Essays*, Cambridge, MA: MIT Press.

Frascolla, P. (1994), *Wittgenstein's Philosophy of Mathematics*, Londres: Routledge.

Frege, G. (1892), "Sobre sentido y referencia", en Valdés, L. (ed.), *Ensayos de semántica y filosofía de la lógica*, Madrid: Tecnos, pp. 84-111.

Gödel, K. (1931), "On Formally Undecidable Propositions of the Principia Mathematica and Related Systems I", en Davis, M. (ed.), *The Undecidable*, Nueva York: Raven Press, pp. 4-38.

Gödel, K. (1934), "On undecidable propositions of formal mathematical systems", en Davis, M. (ed.), *The Undecidable*, Nueva York: Raven Press, pp. 39-74.

Gödel, K. (1944), "La lógica matemática de Russell", en Mosterín, J. (ed.), *Obras completas*, Madrid: Alianza, pp. 313-343.

Gödel, K. (1964), "Postscriptum to Gödel 1931", en Davis, M. (ed.), *The Undecidable*, Nueva York: Raven Press, pp. 71-73.

Kenny, A. (1989), *The Metaphysics of Mind*, Oxford: Oxford University Press.

Kenny, A. (1990), *El legado de Wittgenstein*, Madrid: Siglo XXI.

Kleene, S.C. (1938), "On Notation for Ordinal Numbers", *The Journal of Symbolic Logic* 3: 150-5.

Kleene, S.C. (1943), "Recursive Predicates and Quantifiers", *Transactions of the American Mathematical Society* 53: 41-73.

Kleene, S.C. (1952), *Introduction to Metamathematics*, Amsterdam: North-Holland Publishing.

Marion, M. (1998), *Wittgenstein, Finitism, and the Foundations of Mathematics*, Oxford: Oxford University Press.

Monk, R. (1990), *Wittgenstein: The Duty of Genius*, Nueva York: Free Press.

Mota, S. (2013), "La propiedad de la recursión en el 'Tractatus Logico-Philosophicus' de Wittgenstein y su relación con la Teoría de la Computabilidad y la Lógica Matemática", *Observaciones Filosóficas* 17:

http://www.obervacionesfilosoficas.net/lapropiedaddelarecursion.htm

Mota, S. (2014), "La historia y la gramática de la recursión: una precisión desde la obra de Wittgenstein", *Pensamiento y Cultura* 17: 20-48.

Mounce, H.O. (1981), *Wittgenstein's Tractatus. An Introduction*, Oxford: Blackwell.

Odifreddi, P. (2001), "Recursive Functions: An Archaeological Look", en Claude, C.S., Dinneen, M.J. y S. Sburlan (eds.), *Combinatorics, Computability and Logic*, Londres: Springer-Verlag, pp. 13-31.

Pinto, S. (2002), "El anti-platonismo del *Tractatus* de Wittgenstein", *Theoria* 13: 137-152.

Post, E. (1921), "Introduction to a General Theory of Elementary Propositions", *American Journal of Mathematics* 43: 163-185.

Post, E. (1943), "Formal Reductions of the General Combinatorial Decision Problem", *American Journal of Mathematics* 65: 197-215.

Post, E. (1944), "Recursively Enumerable Sets of Positive Integers and their Decision Problems", en Davis, M. (ed.), *The Undecidable*, Nueva York: Raven Press, pp. 305-337.

Pylyshyn, Z.W. (1991), "Rules and Representations: Chomsky and Representational Realism", en Kashir, A. (ed.), *The Chomskyan Turn*, Oxford: Basil Blackwell, pp. 231-251.

Quine, W.V.O. (1980), *From a Logical Point of View*, Cambridge, MA: Harvard University Press.

Robinson, R. (1947), "Primitive Recursive Functions", *Bulletin of the American Mathematical Society* 53: 925-942.

Rodych, V. (1999), "Wittgenstein's Inversion of Gödel's Theorem", *Erkenntnis* 51: 173-206.

Rodych, V. (2002), "Wittgenstein on Gödel: The Newly Published Remarks", *Erkenntnis* 56: 379-397.

Rodych, V. (2003), "Misunderstanding Gödel: New Arguments about Wittgenstein and New Remarks by Wittgenstein", *Dialectica* 57: 279-313.

Shanker, S.G. (1987), "Wittgenstein versus Turing on Nature of Church's Thesis", *Notre Dame Journal of Formal Logic* 28: 615-649.

Skolem, T. (1923), "The Foundations of Elementary Arithmetic Established by Means of the Recursive Mode of Thought, without the Use of Apparent Variables Ranging over Infinite Domains", en Van Heijenoort, J. (ed.), *From Frege to Gödel. A Source Book in Mathematical Logic, 1879-1931*, Cambridge, MA: Harvard University Press, pp. 302-333.

Soare, R. (1996), "Computability and Recursion", *The Bulletin of Symbolic Logic* 2: 284-321.

Soare, R. (2009), "Turing Oracles Machines, Online Computing, and Three Displacements in Computability Theory", *Annals of Pure and Applied Logic* 160: 368-399.

Tomalin, M. (2011, "Syntactic Structures and Recursive Devices: A Legacy of Imprecision", *Journal of Logic, Language and Information* 20: 297-315.

Turing, A. (1937), "On Computable Numbers, with an Application to the Entscheidungsproblem", en Davis, M. (ed.), *The Undecidable*, Nueva York: Raven Press, pp. 116-151.

Wittgenstein, L. (1922), *Tractatus Logico-Philosophicus*, London: Routledge.

Wittgenstein, L. (1958), *Philosophical Investigations*, Oxford: Blackwell. (Traducción castellana: *Investigaciones filosóficas*, Barcelona: Crítica, 1988.)

Wittgenstein, L. (1974). *Philosophical Grammar*, Oxford: Basil Blackwell. (Traducción castellana: *Gramática filosófica*, México: UNAM, 1992.)

Wittgenstein, L. (1975a). *Philosophical Remarks*, Oxford: Blackwell. (Traducción castellana: *Observaciones filosóficas*, México: UNAM, 1997.)

Wittgenstein, L. (1975b), *Wittgenstein's Lectures on the Foundations of Mathematics, Cambridge, 1939*, Chicago: University of Chicago Press.

Wittgenstein, L. (1978), *Remarks on the Foundations of Mathematics*, Oxford: Blackwell.

Wrigley, M. (1977), "Wittgenstein's Philosophy of Mathematics", *Philosophical Quarterly* 27: 50-59.

ARTÍCULO 4

Mota, S. (2015). ¿Qué es un algoritmo? Una respuesta desde la obra de Wittgenstein, Éndoxa. Series Filosóficas, 36, 317-328.

¿QUÉ ES UN ALGORITMO? UNA RESPUESTA DESDE LA OBRA DE WITTGENSTEIN

WHAT IS AN ALGORITHM? A RESPONSE BASED ON WITTGENSTEIN'S WORK

Sergio Mota*
Universidad Autónoma de Madrid

Resumen: En este trabajo trato de analizar el debate en torno a la pregunta "¿qué es un algoritmo?". En la actualidad hay dos grandes enfoques representados por Y. Gurevich, quien considera que un algoritmo es una *máquina de estados abstracta*, y por Y. Moschovakis, quien considera que un algoritmo es un *recursor*. Mi propuesta es que ambas respuestas a la pregunta pueden subsumirse bajo la concepción de un algoritmo como una serie de formas, propuesta que captura la definición general de algoritmo comúnmente empleada.

Palabras clave: algoritmo, recursor, máquina de estados abstracta, Wittgenstein.

* Doctorando en el Departamento de Psicología Básica, Campus de Cantoblanco, 28049, Madrid (España), sergio.mota.v@gmail.com.

ÉNDOXA: Series Filosóficas, n.º 36, 2015, pp. 317-328. UNED, Madrid

ABSTRACT: In this paper, I will try to analyze the debate on the question "What is an algorithm?" Currently, there are two main approaches represented by Y. Gurevich, who considers that an algorithm is an *abstract state machine*, and by Y. Moschovakis, who thinks that an algorithm is a *recursor*. My proposal is that both approaches to the question can be subsumed under the notion of an algorithm as a series of forms. This proposal captures the general definition of algorithm that is commonly employed.

KEYWORDS: algorithm, recursor, abstract state machine, Wittgenstein.

1. Introducción

En el *Decimocuarto Congreso Internacional de Lógica, Metodología y Filosofía de la Ciencia*, Moshe Vardi (2012) presentaba dos posibles respuestas a la pregunta "¿qué es un algoritmo?". Esas dos posibles respuestas estaban representadas por las posturas de Yuri Gurevich y Yannis Moschovakis.[1]

Por tanto, la pregunta es: *¿es un algoritmo una máquina de estados abstracta o es un recursor?* Parece necesario establecer qué definición es primaria, puesto que, matemáticamente hablando, un recursor puede modelar una máquina de estados abstracta y una máquina de estados abstracta puede modelar un recursor. Vardi propone una *dualidad algorítmica*, concluyendo que un algoritmo es tanto una máquina de estados como un recursor y que ninguna de ellas describe totalmente lo que es un algoritmo, siendo esta dualidad un principio fundamental de la ciencia de la computación. Veremos en qué medida esto es así.

En la siguiente sección presento una breve contextualización histórica, para, en la tercera sección, presentar las concepciones de Gurevich y Moschovakis.

En la cuarta sección, presento una respuesta a la cuestión basada en la obra de Wittgenstein.

[1] Vardi comienza con la siguiente pregunta: "¿No respondió ya Turing a esta cuestión?" Es decir, ¿no es *la* respuesta que un algoritmo es una máquina de Turing? Ciertamente, una máquina de Turing caracteriza la noción de algoritmo, pero no es adecuado decir que *la* respuesta, que la *única* (y *correcta*) respuesta sea que un algoritmo es una máquina de Turing. Una máquina de Turing es *una* instancia del concepto de algoritmo, no *la única* instancia.

2. Breve contextualización histórica

Es necesario exponer, siquiera brevemente, los avances llevados a cabo en la denominada teoría de la computabilidad, una rama de la lógica matemática. Como es bien sabido, la noción de algoritmo fue central durante los años 1930 en adelante, cuando la teoría de la computabilidad (antes llamada "teoría de las funciones recursivas") sufrió una expansión considerable. Diferentes autores son reconocidos como los responsables de dicha expansión, algunos considerados más relevantes o prominentes en el campo que otros. Por ejemplo, ocupan un papel preponderante Kurt Gödel, Alonzo Church o Alan Turing, mientras que otros autores también contribuyeron notablemente, como puedan ser Stephen Cole Kleene o Emil L. Post.

Así, todas las propuestas o sistemas formales establecidos a partir de los años 30 y 40 del siglo pasado son de gran relevancia para, entre otras cuestiones, el estudio de la computabilidad y para caracterizar la noción de algoritmo. Es decir, todas las propuestas llevadas a cabo por los autores antes citados tratan de caracterizar formalmente la noción de algoritmo, esto es, de procedimiento efectivo, o también procedimiento computacional o generacional, o procedimiento de decisión. De este modo, pretendían ir más allá de la noción intuitiva de algoritmo, la cual cambió cuando Hilbert, a comienzos del siglo XX, formuló el llamado 'problema de la decisión' (*Entscheidungsproblem*), que consistía en establecer un procedimiento de decisión –(*Entscheidungsverfahren*)– efectivo que permitiera determinar o decidir, en un número finito de operaciones, si una expresión o fórmula bien formada es o no válida. Obviamente, esto se relaciona con su *programa finitista*, por medio del cual pretendía probar la consistencia de la aritmética. Dicho problema fue mostrado irresoluble por varios de los autores citados anteriormente, como Gödel, Church, Turing o Post (véase Davis, 1965, 1982 para más detalles).

Así, son de crucial relevancia los avances de Church (1932) con respecto al cálculo-λ y su "primera tesis", que establecía que una función es efectivamente computable (calculable) si y sólo si es definida en el cálculo-λ (v.gr. λ-definible). Posteriormente, con la aparición de las funciones recursivas generales (Gödel, 1934), Church (1936) presentó una segunda versión de su tesis, en la que establecía que una función es efectivamente calculable si y sólo si es recursiva.[2] Es

[2] Una función *f* está definida por recursión cuando cada (nuevo) valor (de *f*) se define o especifica haciendo uso sus propios valores previamente computados para argumentos meno-

bien sabido que Gödel no estaba conforme con ninguna de las versiones de la Tesis de Church (cf. Soare, 2009). Además, según han indicado Soare (2009) o Sieg (2006), parece que la identificación hecha por Church (1936) entre funciones computables y funciones recursivas generales adolecía de errores.

Por otro lado, e independientemente de Church, Turing (1936) estableció lo que ahora se conoce como la Tesis de Turing, que indica que una función es (efectivamente) computable si y sólo si es computable por una máquina de Turing. Como señala el propio Gödel (1964), el trabajo de Turing resultó más satisfactorio que el realizado por Church, pues como el propio Gödel señala (pp. 71-72), al trabajo de Turing se debe una adecuada definición del concepto general de sistema formal. Además, como aprecia Gödel (Ibid.), también a Turing es debido un análisis del concepto de procedimiento mecánico (alias "algoritmo" o "procedimiento computacional"). Esto propició, en último término, que aceptara la tesis de Church, toda vez que se reformuló en términos de la tesis de Turing-Church.[3]

En general, se entiende que, lejos de ser totalmente nuevas estas aportaciones, son más bien la continuación de trabajos anteriores, como los de Dedekind, Frege, Russell o Hilbert, por citar algunos. Sin embargo, no se ha prestado suficiente atención, en mi opinión, a la importancia que tiene la noción formal de *forma general de serie de formas* que Wittgenstein introdujo en el *Tractatus* en 1922. Como ejemplo diré que Wittgenstein presenta un sistema formal muy similar

---

res (cf. Cutland, 1980). Siguiendo a Kleene (1943), todas las clases de funciones recursivas, esto es, las primitivas, las generales y las parciales, se construyen haciendo uso de los esquemas I-III, que aluden a la función sucesor, a las funciones contantes y a las funciones identidad como funciones iniciales, y de los esquemas IV (que es el esquema de substitución), y V (que es el esquema de definición por recursión). Bajo este esquema caen las definiciones por recursión de una variable, con parámetros y de doble recursión, aplicada a dos variables simultáneamente, como *diferentes versiones* de tal esquema. Por otro lado, el esquema VI hace referencia al esquema de *minimalización*. La clase de las funciones recursivas generales se cierra bajo I-VI, teniendo que cumplir la función definida mediante el esquema VI la exigencia de que tiene que tener al menos una solución. Al relajar esta exigencia tenemos las funciones recursivas parciales, también definidas mediante I-VI, pero en este caso, una función definida mediante VI no tiene que tener necesariamente una solución (véase Torretti, 1998, para más detalles).

[3] Desde mi punto de vista, y dado que fue Kleene y no Church quien identificó la clase de las funciones computables con la clase de las funciones recursivas parciales –que incluye a las recursivas generales y a las recursivas primitivas como totales–, sería más adecuado formular la tesis de *Turing-Kleene*, la cual se podría usar en los términos en los que se emplea la tesis de *Turing-Church*.

al de Post (1921), mediante el cual se podía determinar, dada una proposición, en un número finito de pasos, si ésta era o no una proposición de la lógica proposicional (v.gr. una tautología o verdad lógica). Sin embargo, la noción lógica de *forma general de una serie de formas* es una noción más general que puede constituir la definición formal misma de algoritmo, tal y como intentaré mostrar más abajo.

3. Gurevich y las máquinas de estados abstractas Vs. Moschovakis y los recursores

Presentado este marco general, me centraré, primero, en la aportación de Gurevich, quien parece señalar que un algoritmo es una máquina de estados abstracta –como por ejemplo una máquina de Turing– (véase Blass y Gurevich, 2007; Gurevich, 2011). Así, Gurevich (2000) está próximo a la siguiente tesis: *todo algoritmo secuencial puede ser, paso a paso, simulado por una máquina de estados abstracta apropiada*.

Un algoritmo secuencial A puede definirse como sigue $[A_I, A_n, {}_{TA}A_n]$, donde A_I está por el estado inicial, A_n está por un estado arbitrario y ${}_{TA}A_n$ es el estado que le sigue por medio de la aplicación de map_{TA} (una operación de transformación) a A_n. Así, la serie de formas (usaré desde el principio esta expresión, aunque su aclaración y justificación se verá más abajo) $A_0, A_1, A_2,\ldots A_{n+1}$ puede definirse recursivamente como sigue: $A_0 = A_0$ Def; $A_{n+1} = map_{TA}(A_n)$ Def. En este sentido, las funciones que cambian de un estado de un algoritmo dado a otro estado se denominan funciones dinámicas, y un ejemplo de ello lo constituye la máquina de Turing (Gurevich, 2000, p. 9). En todo caso, un algoritmo es, siguiendo a Gurevich (op. cit., p. 15), un objeto A que satisface los postulados de secuenciación temporal, de estados abstractos y de exploración delimitada.[4]

[4] ¿Qué pasa con los algoritmos interactivos? ¿Cambia algo? Sí, la manera en que se ejecutan. Así, cómo se ejecuta un algoritmo no interactivo se representa mediante la serie $A_1, A_2, A_3,\ldots A_{n+1}$; mientras que cómo lo hace un algoritmo interactivo se representa mediante la serie $X_1, X'_1, X_2, X'_2, X_3, X'_3,\ldots$, donde el algoritmo hace los movimientos X y el entorno hace los movimientos X'. Así, la noción formal de serie de formas (ver *infra*) puede aplicarse en este caso también, de tal manera que X_0 representa el estado inicial mientras que X_1 es igual a $map_{TA}(X_0)$ ($X_1 = map_{TA}(X_0)$ Def.). Por su parte, X'_1 se obtiene a partir de X_1 por una acción del ambiente y así sucesivamente.

¿Qué hay en relación con los algoritmos no deterministas? Un algoritmo no determinista es aquel en el que dado un estado cualquiera, el algoritmo tiene varias alternativas de ejecución. Con ligeras, por no decir mínimas, modificaciones, en este caso también se mantienen

Sin embargo, como he indicado más arriba, esta no es la única identificación realizada, pues hay otro autor importante en este campo, Yannis Moschovakis, que ha hecho una identificación diferente, según la cual, un algoritmo es un recursor, esto es, una descripción recursiva construida tomando como base un conjunto de operaciones tomadas como primitivas.

Cuando Moschovakis (1998) nos habla *sobre la fundamentación de la teoría de los algoritmos,* indica que un recursor modela la estructura matemática de los algoritmos; esto es, que un algoritmo es una definición recursiva, mientras que las máquinas abstractas modelan las implementaciones (cf. Moschovkis, 2001; Moschovakis y Paschalis, 2008).

Una máquina de estados abstracta, como una máquina de Turing, es considerada por Moschovakis (1998) como un iterador (como opuesto a un recursor: mientras que éste es una definición matemática de un algoritmo, aquél modela su implementación). Desde mi punto de vista, *iteradores* y *recursores* pertenecen a niveles de análisis diferentes. Así, una máquina abstracta, como la máquina de Turing, se puede definir recursivamente, mientras que su implementación es iterativa. Comencemos, con la definición de 'iterador':

Para dos conjuntos Y y Z un iterador $iter_f$: $Y^{\curvearrowright}Z$ es del tipo (i, S, σ, T, o), donde i es el input tomado por la máquina, S es un conjunto no vacío de estados, σ es la función de transición de un estado a otro, T es el es el estado final y o es el

---

los tres postulados de secuencia temporal, de estados abstractos y de exploración delimitada, aunque este último postulado puede no ser requerido, quedando la definición como sigue: un algoritmo es un objeto que satisface los postulados de secuenciación temporal y de estados abstractos (véase Gurevich, 2000, p. 26 para más detalles).

Con respecto a los algoritmos paralelos, como opuestos a los secuenciales, Blass y Gurevich (2007) postulan la siguiente tesis: todo algoritmo paralelo se comporta de modo equivalente a una máquina de estados paralela. En este sentido, y sin entrar en mayores detalles, un algoritmo paralelo reúne los tres postulados mencionados en relación con los algoritmos secuenciales a los que se les añade otra serie de postulados (véase Blass y Gurevich, 2007, para más detalles). En todo caso, para Gurevich, un algoritmo *es* una máquina de estados abstracta. Así, cualquier algoritmo, secuencial o paralelo, se comporta de manera equivalente a una máquina de estados abstracta, lo que supone que, en último término, es necesario expresar un estado inicial, un estado de la secuencia y una operación que permita pasar de un estado a otro, esto es, una operación de transformación (Blass y Gurevich, 2007, p. 27). Por tanto, no es extraño que se pueda concebir un algoritmo paralelo trabajando de modo secuencial; esto es, pasando de una fase k a una fase k+1. Así, de acuerdo con la tesis de la máquina de estados abstracta, un algoritmo es *una máquina de Gurevich* (de estados abstracta).

output o valor que devuelve la máquina. De este modo, dado un input *x*, $x \in X$, la secuencia de estados se puede definir recursivamente como sigue: el caso base es $S_0(x) = x$ Def., $S_{n+1}(x) = S_n(x)$, si $S_n(x) \in T$ (*i.e.*, si es el estado final) Def.; $S_{n+1} = \sigma(S_n(x))$ (*i.e.*, si la secuencia de estados continua) Def. El ser un iterador no tiene que ver con que un algoritmo pueda o no definirse recursivamente, más bien tiene que ver con la consideración de tales modelos como modelos de implementación (*i.e.*, de computación/computabilidad).

Consideremos ahora qué es un recursor. Aunque Moschovakis lo expone en varios artículos (véase 2001; Moschovakis y Paschalis, 2008), en un artículo ya clásico, (Moschovakis, 1998) lo define como sigue:

Un recursor $\alpha: X \curvearrowright W$, desde un conjunto parcialmente ordenado (*partial ordered set*; i.e., *poset*) X a un conjunto W, es una tripla del tipo (D, map_{TA}, V), donde D es el dominio de α (un conjunto parcialmente ordenado inductivo), map_{TA} es la función de transición de α, y V es el valor obtenido. De este modo, Moschovakis (2001) indica que los algoritmos son definiciones recursivas o, dicho de forma similar, la teoría de los algoritmos es la teoría de las ecuaciones recursivas.[5]

Creo que, de manera más o menos explícita, es posible ver cómo tanto una máquina de estados como los recursores pueden hacer uso de definiciones recursivas. Veamos ahora qué es lo que podemos decir desde la obra de Wittgenstein.

[5] Un ejemplo de una función definida por recursión es la función adición o suma, una función recursiva primitiva: $a+0 = a$ Def., $a+(b+1) = (a+b)+1$ Def. Un ejemplo de función recursiva general es: $\varphi(0,y) = \psi(y)$ Def., $\varphi(x+1, 0) = \chi(x)$ Def., $\varphi(x+1, y+1) = \varphi(x, \varphi(x+1, y))$ Def. Como puede verse en Kleene (1952) el esquema V presenta dos versiones, aludiendo a un esquema con una variable sin parámetros (Va), o a un esquema con parámetros (Vb). Como muestra Kleene (1943), el esquema de definición por recursión (V) es también empleado para definir la clase de las funciones recursivas generales (y parciales). Dicho esto, creo que una definición por doble recursión, como la de Gödel (1934), no hace uso de un esquema distinto del esquema V, sino que más bien hace uso de otra *versión* del esquema V, aquel que permite aplicar la inducción sobre dos variables simultáneamente (p. ej. Vc). Así, se dispone de Va, Vb, Vc…

4. Una respuesta desde la obra de Wittgenstein

En este apartado expondré primero la noción de algoritmo que se puede encontrar en textos sobre los fundamentos de la informática (por ejemplo en Fernández y Sáez, 1987), o en artículos especializados (por ejemplo Shanker, 1987), para después ver la relación entre estas propuestas y el concepto formal de *forma general de una serie de formas*. Ello me permitirá formular las dos propuestas analizadas en el apartado precedente en estos términos.

Siguiendo a Fernández y Sáez (1987, pp. 311-312), hay diferentes definiciones de 'algoritmo'. Por ejemplo, Shanker (1987, p. 632) indica que un algoritmo puede entenderse como una secuencia definida de reglas (operaciones) que especifica cómo producir un resultado (output) desde un input dado en un número finito de pasos.

En consonancia con esta definición, Fernández y Sáez (1987, p. 313) presentan la siguiente cuádrupla, que expresa la definición de algoritmo desde la teoría de conjuntos: $A = <Q, E, S, F>$, donde Q es el conjunto de todos los elementos simples y K-fórmulas que pueden describir el cálculo, E es un subconjunto de Q que hace referencia a los datos de entrada, S también es un subconjunto de Q cuyos elementos son los resultados y F se refiere a la regla del cálculo. Esta regla, a partir de un elemento q_0, genera una sucesión $q_1, q_2, q_3,\ldots$ tal que: $q_{n+1} = F(q_n)$, donde $n \in N$, $q_0 \in E \subset Q$.

Creo que se aprecia con claridad la relación entre esta exposición y la presentada en el apartado precedente, con respecto, por ejemplo, a la secuencia de estados de una máquina abstracta (como la de Turing). Pues bien, en 1922, Wittgenstein, en el *Tractatus Logico-Philosophicus*, definió el concepto formal de *forma general de una serie de formas*.[6] Un concepto formal es, o está expresado por, una variable (4.127): "Toda variable es el signo de un concepto formal" (4.1271). La explicación del uso de esa variable define el concepto formal. De este modo "[u]n concepto formal está ya dado en cuanto se da un objeto que cae bajo él" (4.12721). En 4.1273, Wittgenstein indica que "[e]l término general de una serie de formas sólo puede expresarse mediante una variable" (cf. 4.1272, 4.1252, 4.126).

[6] Una serie de formas es una serie ordenada por relaciones (o propiedades) internas (en oposición a externas o contingentes) –véase Wittgenstein (1922, 4.1252).

¿Cuál es el término general de una serie de formas? En 5.2522, Wittgenstein escribe que el término general de una serie de formas *a*, *O'a*, *O'O'a*,… es [*a*, *x*, *O'x*], donde *a* es el comienzo de la serie, *x* es un término de la serie generado, y *O'x* es el resultado de aplicar *O'* a un valor previamente computado *x*.

Queda, pues, analizar cómo la forma general de una serie de formas muestra y recoge lo esencial de las formulaciones expresadas en términos de iteradores, recursores y a la definición de algoritmo dada por Fernández y Sáez.

Dicho esto, un algoritmo es una serie de formas expresada por medio de la siguiente tripla: [$\tilde{\sigma}$, $\tilde{\varepsilon}$, $\acute{O}(\tilde{\varepsilon})$], donde $\tilde{\sigma}$ expresa un input o un conjunto de ítems iniciales, un dominio en el que no es necesario que todos los ítems que se toman como argumentos estén definidos; esto es, que les corresponda un valor. $\tilde{\varepsilon}$ está por un subconjunto de esos ítems iniciales, un subconjunto del dominio o de los datos de entrada. Finalmente, $\acute{O}(\tilde{\varepsilon})$ indica tanto un valor o resultado, como la operación de transición que permite pasar de un valor a otro, expresada por medio de $\acute{O}(\ )$. Es claro que una secuencia como $\tilde{\varepsilon}$, $\acute{O}^{(n)}(\tilde{\varepsilon})$, $\acute{O}^{(n+1)}(\tilde{\varepsilon})$… captura una secuencia de estados como $S_0(\tilde{\varepsilon})$, $S_n(\tilde{\varepsilon})$, $S_{n+1}(\tilde{\varepsilon})$,… Así, el paso de $\tilde{\varepsilon}$ (=$\acute{O}^{(0)}(\tilde{\varepsilon})$) a $\acute{O}(\tilde{\varepsilon})$ muestra la transición de un estado a otro y, por tanto, que el conjunto de estados no está vacío. Además, ambos pueden instanciar el estado final. De este modo, un 'estado', que no tiene por qué tener ningún sentido psicológico, es una instancia de 'forma', una noción lógico-matemática.

La tripla recién expuesta recoge lo esencial, lo común, a las diferentes propuestas para definir un algoritmo. Evidentemente, puede ser expresada por medio de una definición recursiva (cf. Wittgenstein, 1974, 36):[7]

$\acute{O}^{(0)}\tilde{\varepsilon} = \tilde{\varepsilon}$ Def.;

$\acute{O}^{(n)\prime}\tilde{\varepsilon} = \acute{O}'\acute{O}^{(n-1)}\tilde{\varepsilon}$ Def.

Así, este análisis muestra, en mi opinión, que no hay tal dualismo algorítmico. Es decir, estoy de acuerdo con Vardi en que ninguna de las dos respuestas, en términos de máquinas abstractas o en términos de recursores, describe totalmente lo que es un algoritmo, pero no comparto que esta dualidad sea un principio

[7] Un ejemplo: la forma general de una serie de formas '[0, ξ, ξ + 1]', expresa una regla gramatical del tipo 'a+0 = a; a+ (ξ + 1) = (a+ξ)+1'. Éste es un ejemplo claro de recursor.

fundamental de la ciencia de la computación. Como puede apreciarse, tanto una máquina como un recursor no son sino instancias de una serie de formas, estando sus términos (o estados) ordenados por relaciones –o propiedades– internas o formales. Esto es, cada (nuevo) término resulta de aplicar una operación de transición a un término previamente computado.

Por último, quisiera resaltar que un algoritmo, desde la perspectiva wittgensteiniana que defiendo aquí, no debería considerarse un *descubrimiento* matemático de una realidad empírica o platónica, sino, más bien, una *construcción* o *invención* matemática.

Aunque no pretendo establecer que este análisis es el único posible o el único correcto, sí quiero subrayar la relevancia de Wittgenstein para el análisis del concepto formal de algoritmo. Esto, creo, no ha sido suficientemente reconocido, al menos en los textos citados, algunos de los cuales se emplean en el análisis actual del concepto. Creo que la aportación de Wittgenstein es, en todo caso, interesante y relevante.

Mi gratitud a los revisores anónimos por su lectura y útiles comentarios.

Bibliografía

BLASS, Andreas. & GUREVICH, Yuri. (2007). "Abstract state machines capture parallel algorithms: correction and extension". *ACM Transactions on Computational Logic*, *V*, pp. 1-29.

CHURCH, Alonzo. (1932). "A set of postulates for the foundation of logic". *The Annals of Mathematics*, Vol. 33, pp. 346-366.

— (1936). "An unsolvable problem of elementary number theory". *The undecidable*. Ed. Davis, M. New York: Raven Press, pp. 88-107.

CUTLAND, Nigel. (1980). *Computability: an introduction to recursive function theory*. Cambridge: Cambridge University Press.

DAVIS, Martin. (1965). *The undecidable*. New York: Raven Press.

— (1982). "Why Gödel didn't have Church Thesis". *Information and Control*, Vol. 54, pp. 3-24.

FERNÁNDEZ, Gregorio. y SÁEZ, Fernando. (1987). *Fundamentos de informática*. Madrid: Alianza

GÖDEL, Kurt. (1934). "Sobre sentencias indecidibles de los sistemas formales matemáticos". *Obras Completas*. Ed. Mosterín, J. Madrid: Alianza, pp. 167-196.

— (1964). "Postscriptum to Gödel 1931". *The undecidable*. Ed. Davis, M. New York: Raven Press, pp. 71-73.

GUREVICH, Yuri. (2000). "Sequential abstract state machines capture sequential algorithms". *ACM Transactions on Computational Logic*, Vol. 1, pp. 77-111.

— (2011). "What is an algorithm?". *Technical report MSR-TR-2011-116*.

KLEENE, Stephen Cole. (1943). "Recursive predicates and quantifiers". *Transactions of the American Mathematical Society*, Vol. 53, pp. 41-73.

— (1952). *Introduction to metamathematics*. Amsterdam: North-Holland Publishing.

MOSCHOVAKIS, Yiannis. & PASCHALIS, Vasilis. (2008). "Elementary algorithms and their implementations". *New computational paradigms*. Eds. Cooper, S.B., Löwe, B. & Sorbi, A. Springer, pp. 87-118.

MOSCHOVAKIS, Yiannis. (1998). "On founding the theory of algorithms". *Truth in mathematics*. Eds. Dales, H.G. & Oliveri G. Oxford University Press, pp. 71-104.

— (2001). "What is an algorithm?". *Mathematics Unlimited: 2001 and Beyond*. Eds. Engquist B. & Schmid, W. Berlin: Springer, pp. 919-936.

POST, Emil. (1921). "Introduction to a general theory of elementary propositions". *American Journal of Mathematics*, Vol. 43, pp. 163-185.

SHANKER, Stuart. (1987). "Wittgenstein versus Turing on nature of Church's thesis". *Notre Dame Journal of Formal Logic*, Vol. 28, pp. 615-649.

SIEG, Wilfried. (2006). "Gödel on computability". *Philosphia Mathematica*, III, pp. 189-207.

SOARE, Robert. (2009). "Turing oracles machines, online computing, and three displacements in computability theory". *Annals of Pure and Applied Logic*, Vol. 160, pp. 368-399.

TORRETTI, Roberto. (1998). *El paraíso de Cantor. La tradición conjuntista en la filosofía matemática*. Santiago de Chile: Editorial Universitaria.

TURING, Alan. (1936). "On computable numbers, with an application to the Entscheidungsproblem". *The undecidable*. Ed. Davis, M. New York: Raven Press, pp. 116-151.

VARDI, Moshe (2012). "What is an algorithm?". *Communications of the ACM*, Vol. 5(3), p. 5.

WITTGENSTEIN, Ludwig. (1922). *Tractatus logico-philosophicus*. Londres: Routledge.

— (1974). *Philosophical grammar*. Oxford: Blackwell.

Enviado: 30/06/2015

Aceptado: 24/08/2015

ARTÍCULO 5

Mota, S. & Igoa, J. M. (2016). Parsing complex phrases: Recursion, hierarchical structure, and memory load. *For submission*.

Parsing complex phrases: Recursion, hierarchical structure, and memory load

Sergio Mota & José Manuel Igoa

Department of Basic Psychology

Universidad Autónoma de Madrid (Spain)

Abstract

In this paper, we report two experiments in Spanish designed to find out what kind of processes underlie the online parsing of complex phrases with embedded constituents. In Experiment 1, two structures (complex NPs with embedded PPs and complex coordinate NPs), made up of words or pseudowords, were tested in a click-detection paradigm within a sentence comprehension task. This was contrasted, in Experiment 2, with processing of unstructured lists of words (or pseudowords) with the same paradigm. The pattern of RTs to clicks suggests that, unlike processing unstructured lists of words/pseudo-words, both structures (with and without semantic content) share similar processing demands, and consequently, that parsing of both structures may be similarly characterized by means of a recursively defined algorithm at an abstract level of analysis.

Keywords: parsing, hierarchical structure, memory load, recursion.

Resumen

En este artículo presentamos dos experimentos en español diseñados para descubrir qué clase de procesos subyacen al análisis sintáctico en tiempo real de sintagmas complejos con constituyentes anidados. Para ello, se empleó un paradigma de "detección de clicks" con una tarea de comprensión oral aplicada, en el primer experimento, a dos tipos de estructuras, sintagmas preposicionales subordinados insertos en un sintagma nominal y sintagmas nominales coordinados, en oraciones compuestas por palabras o pseudo-palabras. Esto se contrastó, en el segundo experimento, con el procesamiento de listas no estructuradas de palabras (o pseudo-palabras) con la misma tarea. Los resultados sugieren que, a diferencia de lo que ocurre con las secuencias

desprovistas de estructura, ambas clases de estructuras (tanto si contienen palabras como pseudopalabras) comparten demandas de procesamiento parecidas y, por consiguiente, el procesamiento de ambas se puede caracterizar mediante un algoritmo definido recursivamente en un nivel abstracto de análisis.
Palabras clave: análisis sintáctico, estructura jerárquica, carga de memoria, recursión.

1. Introduction

The study of parsing in natural language gravitates between two kinds of issues that have long preoccupied scholars and researchers in the field: one is the discovery of the principles that guide parsing processes and the type of properties they exhibit; the other is the kinds of information and cognitive resources that support their operation, and the way such information types and cognitive resources are used under time constraints. The first issue includes questions like the architecture of the language processor, the relationship between the grammar and the parser –i.e. to what extent grammatical knowledge is directly reflected in parsing processes–, and the role of recursion, a supposedly central property of the human faculty of language, in parsing operations. The second issue is concerned with matters such as the automatic or controlled nature of various processing components, their degree of encapsulation, or the strategic use of attention and memory resources in language processing.

Our aim in the current study is to bring together both issues by means of an experimental inquiry of the processing of two distinct but related complex structures in Spanish, namely, two types of noun phrases (NPs) with embedded constituents. In so doing, we will look for possible differences between both structures in terms of processing demands, which will lead us to consider the role recursion in the explanation of parsing processes.

The first step of this inquiry is to find a proper characterization of the notion of recursion, as it is currently applied in the cognitive science of language. The notion of recursion has aroused the interest of scholars since

the publication of a famous paper by Hauser, Chomsky and Fitch in 2002, where recursion was claimed to be a central property of the human faculty of language. Since then, the role of recursion in language and other cognitive abilities has been a matter of discussion by leading authors in the cognitive sciences (see the criticisms raised by Pinker & Jackendoff, 2005, and Jackendoff & Pinker, 2005, to Hauser et al's paper, and their reply –Fitch, Hauser & Chomsky, 2005–, and subsequent monographs and reviews on the topic by Van del Hulst, 2010, and Corballis, 2011, among many others).

Within Cognitive Science, the property of recursion has been examined from various perspectives and under different disciplines. In Comparative Psychology, the ability of non-human species as distant as tamarin monkeys and starlings to learn various types of grammars, such as finite-state grammars or context-free grammars, has been extensively studied (for discussion, see Fitch & Hauser, 2004; Perruchet & Rey, 2005; Gentner et al., 2006; Rey et al., 2012). Similarly, in Cognitive Neuroscience, it has been claimed that the human brain is capable of computing complex, recursive structures in language, commonly likened to hierarchically self-embedded structures (see Friederici, 2004; Makuuchi et al., 2009; Friederici et al., 2011; Friederich & Friederici, 2013).

As is well known, the notion of recursion was originally formulated in Mathematical Logic and Computability Theory. This sense of recursion, originally found in Dedekind's and Peano's works at the end of the nineteenth century (see Soare, 1996, for details and references), is that of 'definition by recursion' (or recursive definition).[1] In this sense, a function is said to be recursive, or *defined by recursion* in a technical sense, if "it is

[1] This notion, as Mota (2015a, p. 156) has argued, has to be set apart from other related notions. Thus, in Mathematical Logic, Mathematics, and Computer Science, a distinction is drawn between inductive and co-inductive definitions, which provide us with the dual principles inherent in inductive definitions. Thus, inductive definitions provide two different principles that, *a priori*, should not be conflated, namely, the induction principle –which enables us to prove certain properties of the treated objects, such us natural numbers–, and the recursion principle –which guarantees the correct definitions of recursive functions, such as the sum function of example (1) above. Here we will be concerned with the recursive principle (see Barwise & Moss, 1996, for an introduction to co-induction, co-recursion, and co-induction). Therefore, an inductive definition is a different construct from a recursive definition; likewise an inductive definition is a different concept from the induction principle (i.e., mathematical induction).

defined for an argument x by using its own previously defined values (say $f(y)$ for $y < x$) for arguments smaller than x, and also using 'simpler', previously defined functions" (Soare, 1996; Cutland, 1980). A classical example of a recursively defined function with parameters is the sum function, commonly expressed as:

(1) $a+0 = a / a+1 = a'$ Def., [base cases]
$a+(b+1) = (a+b)+1$ Def. [recursive step]

It is common knowledge in the formal sciences that a recursively defined function may or may not be implemented recursively (cf. Abelson et al., 1996, for details). The case presented in (1) refers to a recursive procedure, but in this article we will be mostly concerned with the notion of recursive *process*. A recursive process is usually defined as a process that is executed by an operation that calls itself in a stepwise manner, until the base case is reached. Once this occurs, the process goes on to compute the deferred operations initiated at each step (see Fitch, 2010; Luuk & Luuk, 2011; Lobina, 2014; Mota, 2015a). An example based on the sum function is shown in (2) for computing '2+4' ('2' being the fixed parameter, and '4' the variable):

(2) 2+0 = 2 [base case]
2+4 = (2+3)+1 [recursive step]

2+4 = (2+3)+1,
= ((2+2)+1)+1,
= (((2+1)+1)+1)+1,
= ((((2+0)+1)+1)+1)+1,
……
= 6

Therefore, we may distinguish at least between two levels, namely, the level of definitions –i.e., how the procedure is defined, by recursion in this case–, and the level of processes –i.e., how the procedure is implemented,

which may also be recursive or, for instance, iterative. Such is the case of (2), where the operation '+1' is applied to the last result obtained: 1+1 = 2, 2+1 = 3, 3+1 = 4… 5+1 = 6; 2+4 = $S^4(2)$, where 'S' is the successor function; $a+b = S^b a$.

In contrast to the definition by recursion shown above, the notion of recursion in Cognitive Science is commonly applied to the internal structure of representations (be they linguistic or otherwise). A structure is defined as recursive in this sense whenever a constituent (or structure) A contains another constituent (or structure) B of the same kind (Pinker & Jackendoff, 2005; Moro, 2008). Examples of such structures are phrases or clauses that embed other constituents of the same kind within themselves, as shown in the sentence "*the cabin [in [the mountains]$_{NP}$]$_{PP}$]$_{NP}$ is made of stone*", where the NP "the mountains" is embedded within the NP "the cabin" (cf. Karlsson, 2010a, Luuk & Luuk, 2011), or the so-called *center-embedded sentences*, like "*[the mouse [(that) the cat [(that) the dog chased]$_S$ bit]$_S$ ran]$_S$*", where a relative clause is contained within another relative clause. This second use of 'recursion' is pervasive in current studies in Linguistics and Psycholinguistics. However, it provides an additional meaning, that of *self-embedding*, that is wholly removed from the original one.[2] In fact, as Chomsky (2015, p. 94) famously claimed, recursion is often conflated with *center-embedding*, self-embedding being a special case of center-embedding in which a constituent is contained within another constituent of the same kind. Thus, recursion, when understood as a computational (i.e. generative) mechanism, may yield both center-embedded (or self-embedded) structures and non-embedded structures, such as 'John cut the rope' (see Chomsky 2011). Our emphasis in this paper will be the notion of recursion as applied to parsing processes, irrespective of the structural characterization of their output.[3] Accordingly, the appropriate way to proceed in order to apply the

[2] The notion of self-embedding may also be found at more abstract levels of representation; for instance, Moro (2008) asserts that the scheme [Specifier-[Head-Complement] that characterizes all kinds of phrasal configurations can be applied both to the Specifier and to the Complement, thus yielding (potentially endless) self-embedded structures.

[3] Note that *recursion in performance* does not mean the same as parsing self-embedded (recursive) structures. This is because parsing self-embedded structures may be carried out by using a non-recursive algorithm, as in the famous counting algorithm that is available when processing strings of the form A^nB^n, where *n* stands for a number of

definition of recursion to the performance domain would be to formulate an algorithm that characterizes the processing of different kinds of structures, and to collect empirical evidence relevant for its implementation in online processing.

Most experimental research on parsing has been devoted to study the processing of self-embedded structures (see classical studies such as Yngve, 1960; Holmes, 1973; or Hakes et al., 1976; and more recent works, such as Gibson, 1998; and Karlsson, 2010b). These studies emphasize the role of memory load in processing complex structures as a key issue in the implementation of parsing operations. The underlying rationale is that parsing self-embedded structures requires *recursive* –as opposed to *iterative*– operations, since it involves establishing dependencies between non-adjacent constituents, and hence performing *deferred* operations, whose completion is delayed until the current operation is finished. In other words, a recursive parsing operation so understood entails a greater load in working memory, since at each step, all previously analyzed constituents have to be retrieved in order to proceed. As we mentioned earlier, this is commonly known as the property of 'self-call', since the procedure must call itself at every step of the process. In contrast, iterative processes are not as memory demanding, since they only require that the last analyzed constituent (or generated value) be kept in memory for subsequent processing (see Abelson, et al., 1996; Luuk & Luuk, 2011).

The relevance of working memory limitations in online processing is beyond doubt (Chomsky & Miller, 1963; Hudson, 1996). As Hudson (1996) has argued, requirements of working memory in parsing are related to (1) the number of simple structures the parser keeps active at a given point; and (2) the number of syntactic constituents predicted and completed during parsing. Early evidence for the role of working memory in parsing comes from the observation that memory load increases at the end of major syntactic units, such as clauses, presumably due to the processing costs

---

instances of the symbol of which it is a superscript. The rule A^nB^n generates sequences such as AB, AABB, AAABBB, and so on, which can be characterized as recursive (i.e., self-embedded), if the sequence AAABBB, for instance, is parsed as [A[A[AB]B]B], or as non-recursive, if it is parsed as [AAA][BBB]. The latter structure may be parsed by implementing a counting algorithm such as: "build an array with a given number of A's followed by the same number of B's".

incurred by the parser when integrating information at major syntactic boundaries, and delivering this information to higher-level processes in order to clear working memory for subsequent processing (Carroll & Tanenhaus, 1978; for a comprehensive review, see Fodor, Bever & Garrett, 1974).

The previous considerations allow us to posit that structural complexity (as reflected in the hierarchical layout of sentences and the long-distance dependencies among its constituents) increases parsing difficulty, due to constraints set on working memory during online processing. However, we submit that there is not a direct link from this assumption to the claim that self-embedded structures *require* the use of recursive processes. There are at least two reasons to the contrary. First, recursive processes are costly in terms of memory resources, so whenever there is a cheaper, non-recursive (e.g. iterative) procedure available, it would seem more sensible to use it instead (Stabler, 2009)[4]. Second, besides this 'practical' objection, there is the conceptual concern that recursion is a property of formal procedures. Therefore, to claim that a given process is recursive, at least according to the original sense of the term, amounts to asserting that a recursive rule can be assigned to it, or that it can be characterized by means of an abstract processing algorithm, which is different from its implementation in real time (or, for that matter, in neural structures and circuits).

In the next section, we will report two experiments (each composed of two subexperiments) with the 'click-detection' paradigm and spoken sentences in Spanish, intended to measure processing load at two different sites in the input utterance. Response times to distractor stimuli are taken to reflect cognitive load during online processing. In the introduction to the experiments, we will describe a recursive algorithm characterizing the rules that are supposedly applied when processing the sentences used therein.

[4] A similar line of reasoning applies to the generation (*not* processing) of both embedded and non-embedded structures, insofar as they can be generated by means of iterative procedures (cf. Luuk & Luuk, 2011; Mota, 2015a, 2015b). In the Minimalist Program of generative linguistics, generative procedures have been reduced to the single operation *Merge*, which recursively generates syntactic objects, but proceeds iteratively by applying *n* times the same operation to the last generated output (see Chomsky, 2007, 2008; Mota, 2015a, 2015b).

2. The experiments

Two experiments were run in the present study: the first one used spoken sentences with two alternative structures: (1) sentences with a complex NP (containing 3 embedded PPs) in subject position of a subordinate complement clause (henceforward the 'embedded-PP condition'); and (2) sentences with a coordinate NP (containing 4 NPs) in the same subject position of a subordinate complement clause (henceforward the 'coordinate-NP condition'). The second experiment contained spoken lists of 4 semantically related nouns (or 4 pseudonouns) introduced by a short preamble. The same words/pseudowords were used in both experiments.

In both experiments, participants were engaged in a dual task: they had to listen to the sentences for comprehension (Experiment 1), or to the word/pseudoword list in a probe recognition task (Experiment 2), while monitoring an auditory stimulus (a tone) inserted somewhere in the audio file and executing a manual response as soon as they heard the tone.

Materials in Experiment 1 were spoken sentences consisting of a short main clause plus a subordinate complement clause with a complex NP in subject position. Head nouns in this subject NP were words in Subexperiment 1A, and pseudowords in Subexperiment 1B. Both sentences with words (1A) and pseudowords (1B) were parseable strings, and contained interpretable propositions, though devoid of lexical meaning in Subexperiment 1B. The same contrast between words and pseudowords was used in Experiment 2, which yielded two corresponding subexperiments (2A: words; 2B: pseudowords).

We introduced this 'semantic' contrast in order to test a situation in which parsing decisions would not be biased by lexical meanings. Accordingly, we might expect to find more neutral, syntactically-driven operations in the case of sentences with pseudowords at the relevant positions. As for Experiment 2, we expected to find an increased difficulty in the recognition of pseudowords, given that they lack a lexical

representation in memory. Table 1 below shows examples of materials from both subexperiments.[5]

Experiment	Condition	Sentence / Pseudosentence
Subexperiment 1A Sentences with words	Embedded-PP	*El conductor vio que [la rueda del remolque del camión de las mudanzas]estaba pinchada* Lit. The driver saw that [the tire of the trailer of the truck of the moves] was punctured *(The driver saw that the trailer of the moving truck had a punctured tire)*
	Coordinate-NP	*Los niños creen que [las motos, los camiones, los trenes y los aviones] son muy grandes.* The kids think that [the scooters, the trucks, the trains, and the planes] are very big.
Subexperiment 1B Sentences with pseudowords	Embedded-PP	*El conductor vio que [la **mita** del **frolador** del **fustio** de las **gabrinas**] estaba pinchada.* The driver saw that [the **mipe** of the **fronker** of the **fustee** of the **gabrins**] was pricked.
	Coordinate-NP	*Ayer vimos que [las **logas,** el **nordal**, la **namira** y el **nofón**] eran muy grandes.* Yesterday we saw that [the **loggles**, the **nordle**, the **narmer** and the **noffen**] were very big.

Table 1. Examples of sentences with words and pseudowords used in Subexperiments 1A and 1B. Complex NPs appear in brackets. Pseudowords are in boldface. English pseudowords in the translated examples have been adapted for illustrative purposes.

As for the materials of Experiment 2, each item consisted of a series of 4 nouns belonging to the same semantic category (Subexperiment 2A), or 4 pseudonouns (Subexperiment 2B), following a brief preamble.[6]

[5] A full list of the materials used in this experiment is provided in Appendix 1.
[6] See full list of materials of this experiment in Appendix 2.

2.1. Experiment 1

The main purpose of Experiment 1 was to compare 'embedded-PP' and 'coordinate-NP' structures, two kinds of complex phrases with supposedly different processing demands. The difference between both structures can be stated as follows. Complex NPs with embedded PPs require parsing hierarchically nested structures in a stepwise fashion, while keeping track of the φ-features (gender and number, in the case of Spanish nouns) of the head noun of the complex NP for subsequent agreement with the sentence predicate (i.e. long-distance checking of number features, in this particular case[7]). This might involve complex, deferred operations with concomitant self-calls every time an embedded noun is encountered.

In contrast, parsing coordinate NPs requires adding a new NP constituent to the NP under construction at every step of the process. This might also involve deferred operations, though of a different kind, since the number feature of the complex NP is given by the plurality of the set denoted by the coordinated NP, and thus feature-checking for agreement is not performed in a distant manner as in the former case. So, at first glance, it might be argued that processing embedded PPs in a complex NP imposes heavier demands on working memory than processing coordinate NPs.[8]

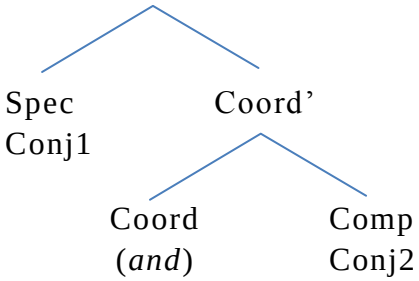
Besides the above mentioned differences between embedded-PP and coordinate-NP structures, we should acknowledge two related syntactic features, relevant for parsing purposes, that are supposedly common to both structures. First, the syntactic (X-bar) configuration is most likely the same in both, namely a binary-branching (and thus hierarchical) configuration. In

[7] In Spanish, subject-verb agreement regularly requires checking of number-features, except for copular or passive sentences (not employed in this study), where the predicate (either a predicate nominative or a verb with a passive participle) also carries gender-features.

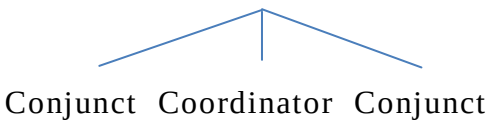
[8] Another possible source of cognitive load in the 'embedded-PP' condition comes from the fact that the head-NP (the first in the sequence of nouns) and its NP-complements embedded in PPs might differ in number, thereby inducing an 'attraction effect', which has been shown to produce agreement errors (i.e. selection or identification of the wrong number feature in the verb) in both comprehension and production tasks, especially when the 'attractor noun' is plural and the head-noun is singular (Vigliocco et al., 1996; Eberhard et al., 2005; Wagers et al., 2014; Lago et al., 2015). This risk was minimized in our study by having our materials match in number in the relevant nouns of the sequence –the first (head-NP) and third nouns– in all but one of the sentences (see Appendix 1).

particular, the purported binary structure of coordinate NPs has been endorsed by various authors (Johannessen, 1998; Camacho, 2003; Zhang, 2010), based on evidence that coordinate NPs have an asymmetric structure. Under this analysis (see 3a below), a coordinate NP is represented as a functional projection (named Coordination Phrase or 'CoordP') that has the conjunction *and* as its head, the first member of the conjunct as its Specifier, and the second member of the conjunct as its Complement. This structure is preferred to the classical 'flat' structure with ternary branching shown in (3b).

(2) a.



b.



There is, however, an alternative analysis of the binary structure of coordinate-NPs, which regards the '*and*-phrase' as an adjunct of the first conjunct (Munn, 1993). Under this analysis, the nouns in a coordinate-NP are independent phrases (i.e., maximal projections, in the parlance of syntactic theory) and are thus simply adjoined to each other, but do not form Specifier-Head-Complement configurations. Thus, they stand in a concatenation relation but bear no specific labels. The choice between these two options ought to have implications for parsing, since a complement-to-head relation between a conjunct and its coordinator entails a different kind of syntactic and semantic dependency among the coordinated nouns in a complex NP to that of a mere concatenative relation.

In view of the above remarks, the status of the relationship between the NPs of both structures under scrutiny remains open. Thus, if the memory demands on agreement prevail we should expect to find processing differences between the embedded-PP and the coordinate-NP structures. On

the other hand, if the parser is only sensitive to the syntactic configuration of phrases, no differences should show up between both structures.

The algorithm proposed to account for online parsing of sentences with the embedded-PPs is shown in (3). It consists of a system of recursive equations, where '$\mathcal{O}$' refers to the parsing operational variable that takes the values of the grammatical operations performed, '*a*' is a variable whose values are syntactic objects, and '$\mathcal{O}^{(n-1)}$'(*a*)' stands for a previously processed syntactic object:

(3) $\mathcal{O}^{(0)}(a) = a$

$\mathcal{O}^{(n)}(a) = \mathcal{O}$'$\mathcal{O}^{(n-1)}$'(*a*)

The sequence of steps that we specify below can help illustrate how the rule proceeds. This description, we insist, should not be understood as an account of the processes occurring in real-time, but an abstract characterization (a set of instructions, so to speak) of how the parser should proceed. In what follows, we formulate a sequential algorithm where each step is recursively characterized (see Gurevich, 2000; Mota, 2015c), as applied to a complex NP with an embedded PP.

For the sake of simplicity, let us consider what happens when the syntactic object 'trailer' is assembled with a previously computed one '*the tire of the*' in order to obtain '*the tire of the trailer*'. This is expressed in concise form as follows:

$\mathcal{O}^{(4)}(the) = \mathcal{O}_{trailer}$'$\mathcal{O}^{(3)}$'(*the*),

the tire of the trailer = {the tire of the}, trailer

$\mathcal{O}_{trailer}$'$\mathcal{O}^{(3)}$'(*the*) = $\mathcal{O}_{trailer}$'$\mathcal{O}_{the}$'$\mathcal{O}^{(2)}$'(*the*),

{the tire of the}, trailer = {{the tire of},the}, trailer

$\mathcal{O}_{trailer}$'$\mathcal{O}_{the}$'$\mathcal{O}^{(2)}$'(*the*) = $\mathcal{O}_{trailer}$'$\mathcal{O}_{the}$'$\mathcal{O}_{of}$' $\mathcal{O}^{(1)}$'(*the*),

{{the tire of},the}, trailer = {{{the tire}}, of}, the}, trailer

There is, yet, a final operation:

$$\mathcal{G}_{trailer}\text{'}\mathcal{G}_{the}\text{'}\mathcal{G}_{of}\text{'}\mathcal{G}^{(1)}\text{'}(the)=\mathcal{G}_{trailer}\text{'}\mathcal{G}_{the}\text{'}\mathcal{G}_{of}\text{'}\mathcal{G}_{tire}\text{'}\mathcal{G}^{(0)}\text{'}(the).$$

{{{the tire}}, of}, the}, trailer = {{{{the}, tire}, of}, the}, trailer[9]

This sequence of steps yields the syntactic structure of the whole syntactic object '*the tire of the trailer*'. Each of these constituents bears its corresponding role as specifier, head and complement of the complex NP. [10]

As for structures with coordinate-NPs, as we mentioned earlier, parsing should proceed in a similar fashion –i.e. the same algorithm would apply– if the parser were only sensitive to their syntactic configuration, which is the same as that of complex NPs with embedded-PPs. The only difference concerning the algorithm would be that the conjunction 'y' (*and*) should be retrieved as coordinator at every step of the procedure (or in each of the recursive rules that are successively applied), that is, every time a new NP is assembled with previously parsed syntactic objects.

Before we move on to report the subsexperiments of this study, we would like to make some remarks concerning the choice of the task. Performance in dual tasks like the one employed in the current study is commonly taken to reflect changes in the allocation of attention or in working memory during online processing of complex stimuli. In particular, response times to distractor stimuli in the click-detection task have been shown to be sensitive to variations in the processing difficulty of structural and semantic information in language (Cohen & Mehler, 1996) and other

[9] Although these operations resemble the generative mechanism *Merge*, it needn't match the procedure followed by it.

[10] An alternative, iterative implementation would result in much less strain in memory load, given that in that scenario, every new syntactic constituent is assembled with the last one computed, not being necessary to retrieve previously processed ones. This is how an iterative algorithm would look like:
$\mathcal{G}^{(1)}(the)$, tire → $\mathcal{G}^{(2)}(the)$ [the, tire]; $\mathcal{G}^{(2)}(the)$, of → $\mathcal{G}^{(3)}(the)$ [the tire of]… $\mathcal{G}^{(n)}(x)$, y → $\mathcal{G}^{(n+1)}(x)$
In this case, an increasing amount of syntactic objects is stored, which, of course, affects memory load (as in the case of a sequence of words; see experiment 2).

domains, such as music (Berent & Perfetti, 1993). Widely known are the results of early psycholinguistic studies showing that response times to clicks are slowed down near or at major syntactic boundaries, or tend to be higher at the beginning than at end of clauses (Cutler & Norris, 1979). More recent research has revealed that clicks tend to be responded faster when located at spoken word boundaries (which are regularly non-physical breaks) than when placed within a word (Gómez et al., 2011). Thus, our reasoning goes, we could take advantage of this measure in order to make an estimation of the cognitive load during processing of complex, hierarchically-structured phrases at various points.

Subexperiment 1A

Participants

Twenty undergraduate and graduate students from the Universidad Autónoma de Madrid (aged 21 to 35) volunteered to participate in this experiment. All of them were native speakers of Spanish, and none had hearing impairments.

Materials

Thirty-two experimental sentences were constructed as described above (see Table 1), 16 with a complex NP with three embedded PPs, and the other 16 with a complex NP with four coordinate NPs. Twenty-four filler sentences with the same structure were added, 12 of each kind. The nouns selected for the complex NPs of the critical sentences were matched for mean frequency and length across conditions [for the embedded-PP condition: mean frequency = 128.31 per million; mean length = 6.16 phonemes; for in the coordinate-NP condition: mean frequency = 116.46 per million; mean length = 6.28 phonemes][11]. Twelve of the 56 items comprising the experiment (21.42%) were followed by a comprehension (yes-no) question to make sure participants remained attentive to the task and to

[11] Source of frequency count: NIM database (Universitat Rovira i Virgili, Tarragona, Spain): Guasch, Boada, Ferré & Sánchez-Casas (2013). http://psico.fcep.urv.cat/utilitats/nim/index.php

filter out participants with too many errors. Comprehension questions are listed in Appendix 1.

All sentences were digitally-recorded by a male speaker and transformed into wav files for auditory presentation. A 100 millisecond tone with a mean frequency of 330.77 Hz and a mean amplitude of 77.96 dB was inserted for the 'click detection' task at two critical positions in the experimental materials: (1) the onset of the last syllable of the first noun in the complex NP (i.e. position N1), and (2) the onset of the last syllable of the third noun in the complex NP (i.e. position N3) (see arrows in 4).

(4) *El conductor vio que [la rueda* ↓N1 *del remolque* ↓N3 *del camión de las mudanzas] estaba pinchada*

Position of the tone was varied in the filler sentences (i.e. placed on the last syllable of the second and fourth nouns of the complex NPs) in order to avoid regularity in tone placement. Each sentence had only one tone, so two versions (lists) of each of the experimental sentences were created.

Design and procedure

A 2×2 factorial design was used, with two within-subject independent variables with two levels each: (1) type of structure (embedded-PP vs coordinate-NP), and (2) tone position (N1 vs N3).

Each subject listened to 56 sentences along the experiment (32 experimental and 24 filler items). Tone position was counterbalanced across sentences, yielding two lists/versions of the experiment. Every sentence appeared only once in each list, with the tone placed at location N1 or N3. Thus, every participant listened to the same sentences, but with different tone locations across the two versions of the experiment. In other words, each participant only listened every sentence once, with the tone either at N1 or at N3 position. All items within each list were grouped in blocks and randomized across and within blocks. Stimuli were administered through headphones by means of the DMDX program (Forster & Forster, 2003) in a dimly lit and quiet room. Participants were instructed to listen carefully to

the sentences and press the 'AltGr' key with their right forefinger as soon as they heard a tone, while keeping track of the sentence meaning. They were told that in some trials they would have to answer a written question presented on the screen by pushing a 'Yes' or 'No' key[12]. The experiment began with six practice trials to familiarize participants with task and materials. RTs to tones and errors in the comprehension questions were recorded by the DMDX software. The experimental session lasted about twenty minutes.

Results

The pattern of RTs to tones are shown in Figure 1. Errors in the comprehension questions were negligible. Reaction time data were trimmed by replacing data points below or above two standard deviations over the mean for each participant by the cutoff points (mean ± 2SD). Only 3.28 percent of data were so replaced.

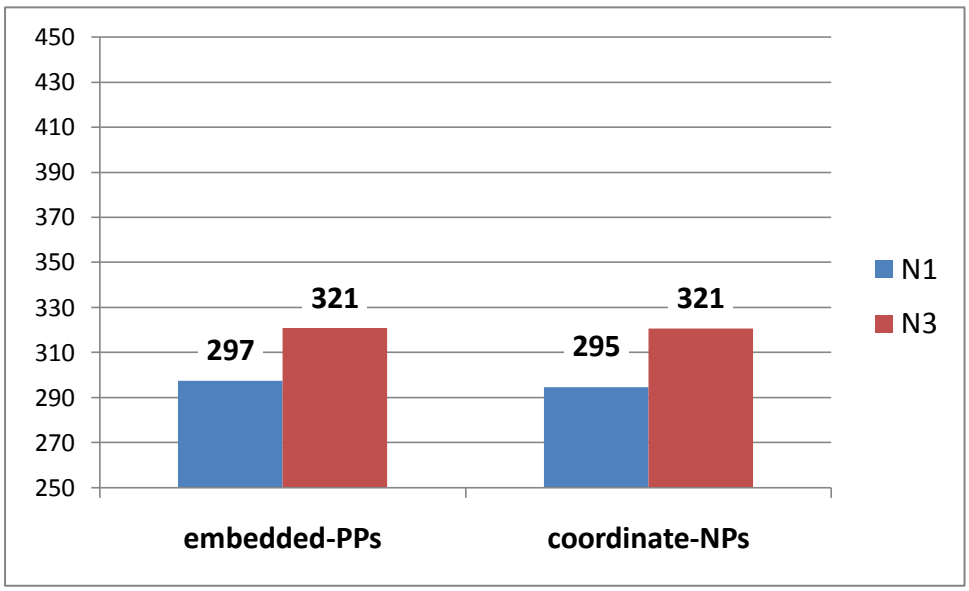


Figure 1. Mean reaction times to tones in sentences with embedded PPs and coordinate NPs as a function of tone position.

[12] As can be seen in Appendix 1, comprehension questions asked about the predicate-argument relation between one of the four nouns in the complex NP and the subordinate verb.

A repeated measures ANOVA with participants and items as random variables was applied on the RT data, yielding the following results: (1) a main effect of tone position which was significant in both participants and items analyses (F_1 (1,19)=12.826, $p<0.005$; F_2 (1,15)=61.933, $p<0.001$); as shown in Figure 1, RTs were significantly faster for tones at N1 position; (2) no main effects of the type of structure or interaction between tone position and type of structure (all F_1 and F_2's < 1). Pairwise comparisons between RTs at positions N1 and N3 revealed a significant advantage for RTs at N1 over RTs at N3 in both cases (for embedded PPs: $t(19) = 2.978$, $p<0.01$; for coordinate NPs: $t(19) = 3.494$, $p<0.005$).

Thus, there appears to be a processing cost associated to the presence of a distractor stimulus when it is located in a constituent that is more deeply embedded or located later in the structure it belongs to. Interestingly, almost exactly the same processing cost accrues for structures with embedded-PPs and with coordinate-NPs. At first glance, these results are compatible with an account based on the view that at N3 position, processing load becomes increased due to deferred operations needed to keep track of the NP under process. However, the lack of differences between the two structures examined calls for an explanation, given the fact that on some accounts, the coordinate-NP structure is thought to be simpler in processing terms. We will come back to these results in the Discussion section.

Subexperiment 1B

Participants

The same twenty participants of Subxperiment 1A took part in this subexperiment.

Materials

The set of experimental and filler sentences used in Subexperiment 1B were modified by replacing the nouns in the complex NPs by phonotactically legal pseudo-nouns in Spanish with the same number of syllables and stress pattern as the original nouns. The number and

proportion of experimental and filler pseudosentences, as well as the length of the items, were identical to those in Subexperiment 1A.

Pseudosentences were digitally-recorded by the same male speaker as in the previous experiment, and tones were inserted in the same positions (N1 and N3, for experimental pseudosentences, and N2 and N4 for the fillers). Two lists were composed for administration of this subexperiment.

Design and procedure

The design and procedure were the same as those of Subexperiment 1a. Comprehension questions were constructed by using the appropriate pseudowords in such a way as to be answerable with a 'Yes' or 'No' response.[13]

Both Subexperiments 1A and 1B were run in the same experimental session, with a short break between them. Participants were randomly administered the two subexperiments in two different orders –either 1A first, then 1B, or the other way around–, with half of the participants following each order. The whole experimental session lasted altogether about 45 minutes.

Results

Mean RTs to tones in the pseudosentences of this subexperiment are depicted in Figure 2. Errors in comprehension questions accounted for less than 10 percent overall. Data points corrected to cutoff values amounted to 4.37 percent of all responses.

[13] See Appendix 1 for comprehension questions.

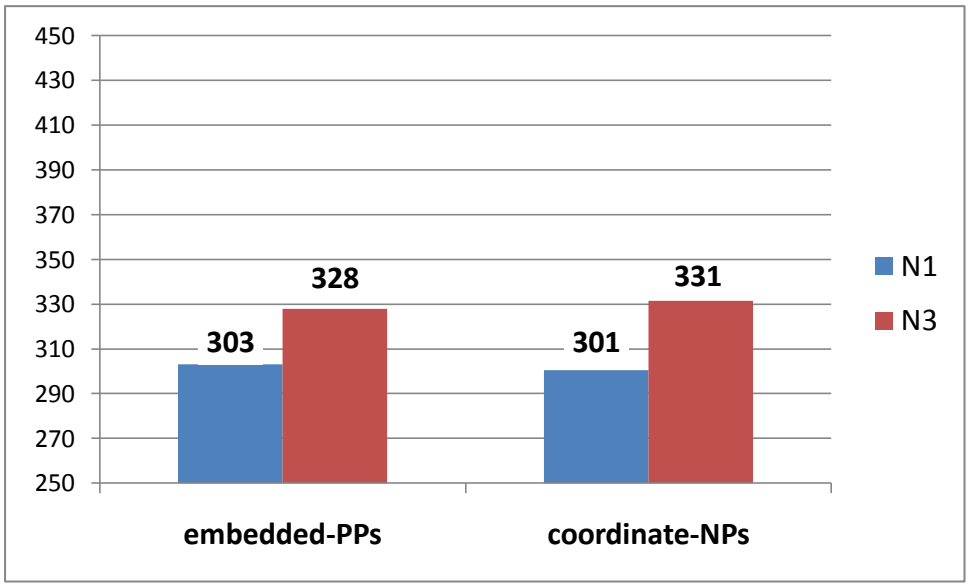


Figure 2. Mean reaction times to tones in pseudosentences with embedded PPs and coordinate NPs as a function of tone position.

A repeated measures ANOVA run on the RT data, with participants and items as random variables, yielded virtually identical results to those found in the previous experiment. There was a highly significant effect of tone position in both participants and items analyses ($F_1(1,19)=19.537$, $p<0.001$; $F_2(1,15)=62.550$, $p<0.001$), with faster RTs for tones located at N1 than at those located at N3, but no effects of type of structure or of the interaction between both factors (again, all F's < 1). Pairwise comparisons between RTs at positions N1 and N3 confirmed the significant advantage of RTs at N1 position in both subordinate ($t(19) = 3{,}429$, $p<0.005$), and coordinate NPs ($t(19) = 4{,}25$, $p<0.001$).

Exactly the same pattern of results obtained with meaningful sentences (Experiment 1A) was replicated with pseudosentences with no lexical-semantic content, though interpretable in terms of thematic role assignment. This provides support to the view that the effects found in these experiments are syntactic in nature. The results of this experiment will be discussed later in conjunction with those of Experiment 1A.

2.2. Experiment 2

This experiment was designed to test whether the effects found in Experiment 1 could be explained in terms of general processing factors, and not as a result of recursive operations during parsing. On this account, two opposite effects may be engendered, depending on which processing factor is considered to be more influential. If memory resources are more decisive in performance, the detection of a distractor tone in the subsidiary task should become more costly (with increasing RTs) as the listener goes through the sentence. Alternatively, processing might also be influenced by a perceptual, serial-position effect, according to which, uncertainty about an upcoming event during processing (i.e. the distractor stimulus) is higher at the beginning of the sentence, resulting in longer RTs, and steadily decreases as the listener moves along the sentence, yielding shorter RTs in posterior locations.

Recent unpublished work using the 'click-detection' paradigm (Lobina et al., 2014) showed that in simple sentences of the form NP-VP-NP, reaction times in the detection of a tone decrease from sentence-initial to sentence-middle positions, and increase again at sentence-final position, in a V-shaped fashion. According to the authors, these results can be explained by a combination of syntactic and perceptual effects. As the parser proceeds along the sentence, there is less material to process, and thus, fewer (structural) expectations and decisions to verify. Therefore, more resources can be devoted to perform the secondary task of click detection. Thus, there is a perceptual, serial-position effect at work, according to which, uncertainty about an upcoming event during processing (i.e. the distractor stimulus) is higher at the beginning of the sentence, resulting in longer RTs, and steadily decreases as the listener moves along the sentence, yielding shorter RTs in posterior locations. However, serial-position effects are reverted at the end of the clause, due to so-called 'wrap-up' effects commonly associated to the integration of syntactic and semantic decisions (Abrams & Bever, 1969; Bever & Hurtig, 1975; Holmes & Forster, 1970).

If memory load, as a general cognitive factor, is more influential, RTs to extraneous stimuli should increase when located later in an array of linguistic items, *irrespective of the structural make-up of the materials*. This demonstration would rule out an account of the results of our former experiments based on recursive processing. In contrast, if perceptual factors take a leading role in the dual task performance, we should expect the opposite, namely, a progressive decline in RTs along the input. The crucial test in either case is whether the pattern of RTs to distractor stimuli holds equally for structured (i.e., sentences) and unstructured materials (i.e., lists of words or arrays of nonlinguistic stimuli). If such were the case, a recursive account of the results of the first two experiments of this study should be abandoned; if not, the recursive explanation could be upheld.

In order to test these alternative hypotheses, we designed a control experiment with the 'click-detection technique, divided into two subexperiments, one with words (Subexperiment 2A) and the other with pseudowords (Subexperiment 2B). Participants listened to strings of words belonging to a given semantic category, or to strings of pseudowords. Their task was to monitor a series of four spoken words/pseudowords in a probe recognition-memory task, and respond to a distractor tone. In 25 percent of trials, they had to report whether the final word/pseudoword was in the previous string.

Subexperiment 2A: Words

Participants

Twenty native speakers of Spanish took part in this experiment. None of them had participated in Experiment 1.

Materials

Twelve sentences containing strings of 4 words taken from the materials of Subexperiment 1A were used as stimuli. Sentences started with a short preamble (e.g. 'The words (pseudowords) are…', 'You should recall…', 'Listen carefully:…'), followed by a sequence of four items (bare nouns in singular or plural form); after a 1200-millisecond break, they heard

either the probe item, or a 1000 Hz tone as signal to advance to the next trial. The strings were constructed by splicing the audio files of the two previous experiments to extract the four nouns from each of the sentences and pseudo-sentences used in the previous experiments, and stringing them together in a natural intonation pattern. The same tone used in Experiment 1 was inserted at the end of the first or the third noun (positions N1 or N3, 50 percent of trials each) of the string in the experimental items. Four filler items of the same kind (strings of words preceded by a short preamble) were added as distractors, with the tone located at positions N2 or N4.

All items of this subexperiment were administered under a single list with 32 items that was split into two parts containing 16 items in each. Each item –experimental or filler– appeared once on each half of the subexperiment, with the distractor tone located at a different position each time. Thus, there were 24 experimental items altogether (12 with tone at N1 and 12 with tone at N3), plus 8 filler items. Each subexperiment began with a list of four practice trials. Probe words appeared in 6 out of 16 trials (37.5% of trials) on each part of the list, and the 1000 Hz tone in the remaining 20 trials (62.5%).

Subexperiment 2B: Pseudowords

Participants

The same 20 participants who took part in Subexperiment 2A did so in this subexperiment.

Materials

Exactly the same procedure as that of Subexperiment 2A was followed to construct the items for this subexperiment, but using pseudowords from Subexperiment 1B. Thus, this subexperiment comprised 12 strings with 4 pseudowords each, with the distractor tone located at positions N1 or N3, plus 4 more strings of 4 pseudowords each, with the tone placed at positions N2 or N4. A single list of 32 items (24 experimental and 8 fillers) was created and split in two halves, with the same distribution of items as in Subexpertiment 2A.

Subexperiments 2A and 2B

Design and procedure

For the purpose of data analysis, a combined design for both sub-experiments was arranged, with 'lexicality' (words vs pseudowords) and position (N1 vs N3) as independent variables. The procedure was the same as that of Experiment 1. Data collapsed across both subexperiments will be shown in the Results section.

Participants were randomly assigned to one of the two subexperiments, which were administered in two possible orders: half of them did the word subexperiment (2A) first, and the other half the pseudoword subexperiment (2B) first. The whole experimental session lasted approximately 20 minutes.

Results

Errors were negligible in the probe task. Corrected data points in the word subexperiment (2A) amounted to a scant 2.6 percent, while in the pseudoword subexperiment (2B), they raised to 3.12 percent. Mean RTs to the tones at critical positions in the experimental stimuli are displayed in Figure 3.

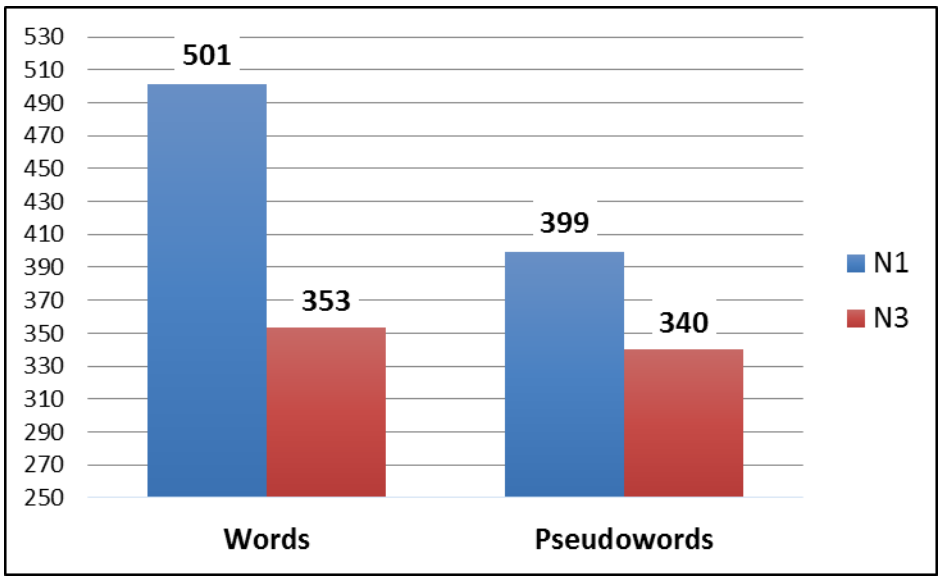


Figure 3. Mean reaction times to tones in word and pseudoword strings as a function of tone position.

A repeated-measures ANOVA run on the RTs, with participants and items as random variables, showed significant effects, in both participants and items analyses, of position ($F_1(1,19)=96,061$ $p<0.001$; $F_2(1,11) = 101.487$, $p<0.001$), lexicality ($F_1(1,19)= 12.263$, $p>0.005$; $F_2(1,11)=57.372$, $p>0.001$), and the lexicality × position interaction ($F_1(1,19)=25,299$ $p<0.001$; $F_2(1,11)=14.088$, $p<0.005$). As shown in Figure 3, tones located at N3 position were responded to significantly faster than those placed at N1 position. Moreover, this difference was much greater for words (148 msec.) than for pseudowords (59 msec.), hence the interaction effect found in the analysis, though both differences were equally significant in the *t* tests (for word targets: $t(19) = 9,977$, $p<0.001$; for pseudoword targets: $t(19) = 4,703$, $p<0.001$).

These results reveal two interesting and complementary consequences: First, they show that when structural constraints are removed from the materials, listeners are more prone to suffer the effects of uncertainty of upcoming events at early locations within a sequence of stimuli regardless of their lexical status. In other words, they suggest that the pattern of responses to distractor tones in the current experiment is more compatible with a perceptually-based account. Moreover, they allow to rule out an account based on sheer memory load of the pattern of response times to tones found in Experiment 1.

On the other hand, the lexicality × position interaction is due to the substantial increase in RTs to tones located at N1 position in word targets, when compared to the other three experimental conditions (see Figure 3). Semantic processing requirements (affecting words, but not pseudowords) might enhance the uncertainty effect induced by the recognition task, and this disturbs more noticeably the recognition of the first member of the series, since the other three elements belong to the same semantic category, which is gradually disclosed as subsequent items come along. Obviously, no comparable effect occurs in strings of pseudowords.

3. Discussion and conclusions

The experiments reported in this paper have shown that listeners take more time to detect a distractor tone embedded within a complex NP in a sentence comprehension task, when the tone is located deeper in the structure, or at a later position within the NP, as compared to tones located at structurally higher (or earlier) positions. This effect appears to be syntactic in nature, since it also occurs in sentences devoid of lexical meaning (see Humphries et al., 2007; Brennan & Pylkkänen, 2012). Furthermore, the opposite pattern of RTs was encountered when the tone was embedded in an unstructured string of words (or pseudowords), which suggests that participants' sensitivity to distractor stimuli is bound to parsing operations, and not sensitive to perceptual factors such as the serial position of the distracter stimulus in the string.

A possible account of these effects is that the attachment of an embedded PP (e.g. '*del camión*' in '*la rueda$_{N1}$ del remolque$_{N2}$ del camión$_{N3}$' –*'of the truck'* in *'the tire$_{N1}$ of the trailer$_{N2}$ of the truck$_{N3}$'*) to the current maximal NP –or DP, to be more precise– (headed by '*la rueda*'–*'the tire'*) entails an increase in processing costs associated to the computation of the current argument (i.e. the whole NP) by retrieving values that have been previously computed for a smaller argument, namely the complex NP/DP that has been parsed so far (i.e. '*la rueda del remolque*' –*'the tire$_{N1}$ of the trailer$_{N2}$'*). By hypothesis, this kind of computation would ensue every time a new constituent is attached to the matrix constituent, and it squares with the recursive definition as applied to mathematical functions and generative procedures (see section 1 above).

A similar line of reasoning may be followed to make sense of the pattern of response times to tones in the 'coordinate-NP' condition. The attachment of an incoming conjunct in complex coordinated NPs (e.g. *'(y) el freno'* in '*el cambio$_{N1}$ (y) el embrague$_{N2}$ (y) el freno$_{N3}$*' –*'(and) the brakes'* in *'the gearbox$_{N1}$ (and) the clutch$_{N2}$ (and) the brakes$_{N3}$'*), regardless of its being complement or adjunct, triggers the retrieval of the NP-conjuncts processed so far (i.e. values of the complex NP previously computed for smaller arguments).

The analysis of coordinate NPs we have just sketched rests on the assumption that these structures are similar to complex NPs with embedded PPs, both sharing an analogous hierarchical configuration. Given this structural similarity, we submit that the parser is subject to the same working memory constraints when processing both kinds of structures. Thus, the parser starts by encoding the first noun of the complex NP, and labeling it as subject of the subordinate clause, which sets off the expectation of a predicate. This expectation should remain in memory through the series of intervening constituents (PPs or NPs) that follow the first NP, until retrieval of the predicate is feasible. This brings about storage-load effects that are enhanced by the structural identity of the intervening constituents, thereby giving rise to a temporary interference that causes RTs to an extraneous tone to increase when the tone is located within one of the intervening phrases (e.g., at position N3). This account of the process is congruent with the assumptions of current parsing models that attribute a significant role to working memory in sentence processing (e.g. Lewis et al., 2006), either as a result of storage-load or by virtue of interference effects.

Whatever difference may lie between the two structures tested in our experiments, such difference did not result in any significant variation in our reaction-time data. This could be due to a lack of sensitivity of RT measures in the tone-monitoring task to this nuanced difference, or to a bias introduced by the comprehension questions used in our experiments. These questions queried about some particular fact or characteristic of one of the four nouns in the sequence of NPs/PPs said in the predicate (see questions in Appendix 1), which might have encouraged participants to retain all four nouns of the complex NPs in our experiments.

Accordingly, the evidence provided by our experiments shows that a series of phrases of the same kind embedded within a complex NP cause a similar strain on working memory when parsing embedded PPs within an NP and coordinate NPs, and this effect appears to be purely syntactic. Given this, we may now ask how far we can go in our interpretation of the effects found in our experiments concerning the putative recursive nature of syntactic processes in language comprehension. In this regard, we may draw

the following conclusions: (1) the pattern of reaction times to tones embedded in complex syntactic structures of the kind used in our experiments suggests that memory load increases as embedded or coordinated constituents accumulate in the course of processing; (2) this may be taken as an indication that the parser is keeping track of encoded constituents so as to perform deferred syntactic operations, like subject-verb agreement, at a later processing stage; (3) if this is the case, the parser's behavior may be described as following a rule (algorithm) that can be characterized as recursive. This last conclusion rests on the assumption that linguistic performance is a type of rule-following behavior. A more fine-grained account of the underlying processes –for instance, to figure out whether processing of material in the storage interval produces storage-load or interference effects, or testing whether or not parsing a given type of structure actually involves carrying out deferred operations– would require more sensitive tasks and materials, in order to arrive at a more satisfactory answer to the problem of recursion in syntactic processing.

References

Abelson, H. & Sussman, G. J. with Sussman, J. (1996). *Structure and interpretation of computer programs*. Cambridge, MA: MIT Press.

Abrams, K. & Bever, T. G. (1969). Syntactic structure modifies attention during speech perception and recognition. *The Quarterly Journal of Experimental Psychology*, 21(3), 280-290.

Mota, S. (2015a). Sobre el concepto de recursión y sus usos, *Praxis Filosófica, 40,* 153-181.

Mota, S. (2015b). ¿Por qué se usa 'recursión' cuando se quiere significar 'auto-inclusión'?: Clarificaciones conceptuales sobre la recursión en el programa chomskiano, *Revista de Lingüística Teórica y Aplicada*, 53, 171-191.

Mota, S. (2015c). ¿Qué es un algoritmo? Una respuesta desde la obra de Wittgenstein, *Éndoxa. Series Filosóficas,* 36, 317-328.

Barwise, K. J. & Moss, L. (1996). *Vicious Circles. On the Mathematics of Non-Wellfounded Phenomena.* Chicago, IL: University of Chicago Press.

Berent, I. & Perfetti, C.A. (1993). An on-line method in studying music parsing. *Cognition*, 46, 203-222.

Bever, T. G. & Hurtig, R. R. (1975). Detection of a nonlinguisic stimulus is poorest at the end of a clause. *Journal of Psycholinguistic Research*, 4(1), 1-7.

Brennan, J. & Pylkkänen, L. (2012). The time-course and spatial distribution of brain activity associated with sentence processing. *Neuroimage*, 60, 1139–1148.

Camacho, J. (2003). *The structure of coordination Conjunction and Agreement Phenomena in Spanish and Other Languages*. Dordrecht: Kluwer.

Carroll, J.M. & Tanenhaus, M.K. (1978). Functional clauses and sentence segmentation. *J. of Speech and Hearing Research*, 21, 693-708.

Chomsky, N. & Miller, G. (1963). Introduction to the formal analysis of natural languages. In R.D. Luce, R.R. Bush & E. Galanter (*Eds.*), *Handbook of mathematical psychology, Vol. II* (pp. 269-322). New York: John Wiley.

Chomsky, N. (1995). *The minimalist program*. Cambridge, MA: MIT Press.

Chomsky, N. (2007). Approaching UG from below. In U. Sauerland & H. M. Gärtner (eds.), *Interfaces + Recursion = Language?* (pp. 1-30). Berlin: Mouton.

Chomsky, N. (2008). On phases. In R. Freidin, C. Otero, & M. L. Zubizarreta (*Eds.*), *Foundational issues in linguistic theory* (pp. 133-166). Cambridge. MA: MIT Press.

Chomsky, N. (2011). Language and other cognitive systems. What is special about language?. *Language Learning and Development, 7*, 263-278.

Chomsky, N. (2015). Some core contested concepts, *Journal of Psycholinguistic Research, 44*, 91-104.

Cohen, L. & Mehler, J. (1996). Click-monitoring revisited: An on-line study of sentence comprehension. *Memory & Cognition*, 24(1), 94-102.

Corballis, M. C. (2011). *The Recursive Mind: The Origins of Human Language, Thought, and Civilization*, Princeton, NJ: Princeton University Press.

Cutland, N. (1980). *Computability: an introduction to recursive function theory*. Cambridge, UK: Cambridge University Press.

Cutler, A., & Norris, D. (1979). Monitoring sentence comprehension. In W.E. Cooper & E.C.T. Walker (eds.), *Sentence processing: Psycholinguistic studies presented to Merrill Garrett* (pp. 113-134). Hillsdale, NJ: Erlbaum.

Eberhard, K.M., Cutting, J.C., Bock K. (2005). Making sense of syntax: Number agreement in sentence production. *Psychological Review*, 112, 531–559.

Fitch, W.T. (2010). Thee meanings of 'recursion': Key distinctions for biolinguistics. In R.K. Larson, V. Déprez & H. Yamakido (eds.), *The evolution of human language*. (pp. 73-90).

Fitch, W.T., & Hauser, M.D. (2004). Computational constraints on syntactic processing in a nonhuman primate, *Science*, 303, 377-380.

Fitch, W.T., Hauser, M.D. & Chomsky, N. (2005). The evolution of the language faculty: clarifications and implications. *Cognition*,97, 179-210.

Fodor, J.A., Bever, T.G. & Garrett, M.F. (1974*). The Psychology of Language: An Introduction to Psycholinguistics and Generative Grammar*. New York: McGraw-Hill.

Forster, K.I. & Forster, J. (2003). DMDX: A Windows display program with millisecond accuracy. *Behavior Research Methods, Instruments, & Computers*, 35(1),116-124.

Friederich, R.M. & Friederici, A.D. (2013). Mathematical logic in the human brain: semantics, *PLOS one*, *8*(1): e53699. doi:10.1371/journal.pone.0053699

Friederici, A.D. (2004). Processing local transitions versus long-distance syntactic hierarchies. *Trends in Cognitive Science*, 8, 245-247.

Friederici, A.D., Bahlmann, J., Friedrich, R., & Makuuchi, M. (2011). The neural basis of recursion and complex syntactic hierarchy, *Biolinguistics,* 5, 87-104.

Gentner, T. Q., Fenn, K. M., Margoliash, D., & Nusbaum, H. C. (2006). Recursive syntactic pattern learning by songbirds. *Nature*, 440, 1204–1207.

Gibson, E. (1998). Linguistic complexity: locality of syntactic dependencies. *Cognition,* 68, 1-76.

Gómez, D.M., Bion, R.A.H., & Mehler, J. (2011). The word segmentation process as revealed by click detection. *Language and Cognitive Processes,* 26(2), 212-223.

Guasch, M., Boada, R., Ferré, P., & Sánchez-Casas, R. (2013). NIM: A Web-based Swiss Army knife to select stimuli for psycholinguistic studies. *Behavior Research Methods, 45, 765-771.*

Gurevich, Y. (2000). Sequential abstract state machines capture sequential algorithms. *ACM Trans. Computational Logic*, 1(1), 77–111.

Hakes, D., Evans, J.S., & Brannon, L. (1976). Understanding sentences with relative clauses. *Memory & Cognition*, 4, 283-290.

Hauser, M., Chomsky, N. & Fitch, T. (2002). The faculty of language: What is, who has it, and how did it evolve?, *Science*, *298*, 1569-1579.

Holmes, V. M. & Forster, K. I. (1970). Detection of extraneous signals during sentence recognition. *Perception and Psychophysics*, 7(5), 297-301.

Holmes, V.M. (1973). Order of main and subordinate clauses in sentence perception. *J. of Verbal Learning and Verbal Behavior*, 12, 285-293.

Hudson, R. A. (1996). The difficulty of (so-called) self-embedded structures. *UCL Working Papers in Linguistics* 8, 283-314.

Humphries, C., Binder, J.R., Medler, D.A., & Liebenthal, E. (2007). Time course of semantic processes during sentence comprehension: an fMRI study. *Neuroimage,* 36(3), 924–932.

Jackendoff, R. & Pinker, S. (2005). The nature of the language faculty and its implications for evolution of language (reply to Fitch, Hauser, and Chomsky), *Cognition*, *97*, 211-225.

Johannessen, J.B. (1998). *Coordination*. Oxford, UK: Oxford University Press.

Karlsson, F. (2010a). Syntactic recursion and iteration In H. van der Hulst, (ed.), *Recursion and Human Language*. (pp. 43-67). Berlin/New York: Mouton de Gruyter.

Karlsson, F. (2010b). Multiple final embedding of clauses. *International Journal of Corpus Linguistics,* 15(1), 88-105.

Lago, S., Shalom, D.E., Sigman, M., Lau, E., & Phillips, C. (2015). Agreement attraction in Spanish comprehension. *Journal of Memory and Language,* 82, 133–149.

Lewis, R.L., Vasishth, S. & Van Dyke, J.A. (2006). Computational principles of working memory in sentence comprehension. *Trends in Cognitive Sciences*, 10 (10), 447-454.

Lobina, D.J. (2014). Probing recursion. *Cognitive Processing,* 15(4), 435-450.

Lobina, D.J., Demestre, J., & García-Albea, J.E. (2014). A re-evaluation of the click-detection technique and data. Unpublished manuscript.

Luuk, E. & Luuk, H. (2011). The redundancy of recursion and infinity for natural language, *Cognitive Processing*, 12, 1-11.

Makuuchi, M., Bahlmann, J., Anwander, A., & Friederici, A.D. (2009). Segregating the core computational faculty of human language from working memory. *PNAS*, 106(20), 8362–8367.

Moro, A. (2008). *The boundaries of Babel*. Cambridge, MA: MIT Press.

Munn, A. (1993). Topics in the syntax and semantics of coordinate structures, Unpublished PhD dissertation, University of Maryland.

Perruchet, P., & Rey, A. (2005). Does the mastery of center-embedded linguistic structures distinguish humans from nonhuman primates?. *Psychonomic Bulletin & Review*, 12(2), 307-313.

Pinker, S. & Jackendoff, R. (2005). The faculty of language: What's special about it?, *Cognition*, *95*, 201-236.

Rey, A., Perruchet, P., & Fagot, J. (2012). Centre-embedded structures are a by-product of associative learning and working memory constrains: Evidence from baboons (*Papio Papio*), *Cognition*, *123*, 180-184.

Soare, R. (1996). Computability and recursion, *The Bulletin of Symbolic Logic*, 2, 284-321.

Stabler, E.P. (2009). Recursion in gramar and performance. Talk presented at the conference ‘Recursion: Structural complexity in language and cognition’. Amherst, Massachusetts, may (reprinted in T. Roeper & M. Speas (eds.), Recursion: C*omplexity in cognition*, 2014, pp 159-177).

Van der Hulst, H. (2010). *Recursion and Human Language*, Berlin: Mouton the Gruyter.

Vigliocco G, Butterworth B, Semenza C. (1995). Constructing subject–verb agreement in speech: The role of semantic and morphological factors. *Journal of Memory and Language*, 34, 186–215.

Wagers MW, Lau EF, Phillips C. (2009). Agreement attraction in comprehension: Representations and processes. *Journal of Memory and Language*, 61, 206–237.

Wirth, N. (1986). *Algorithms and data structures*. Hemel Hempstead, UK: Prentice Hall.

Yngve, V.H. (1960). A model and a hypothesis for language structure. *Proceedings of the American Philosophical Society*, 104(5), 444-466.

Zhang, N. (2010). *Coordination in Syntax*. Cambridge, UK: Cambridge University Press.

APPENDIX 1: LIST OF MATERIALS USED IN EXPERIMENT 1

SUBEXPERIMENT 1A

1.1. Embedded-PPs: List of experimental and filler sentences used in Subexperiment 1A, including comprehension questions and their correct answers.

EXPERIMENTAL SENTENCES

1. Alguien dijo que los bichos de las hojas de las plantas de la terraza estaban muertos.
2. María vio que las agujas del reloj de la fachada de la iglesia estaban rotas.
3. Me pareció que los dibujos del comic de la revista del domingo eran muy graciosos.

   Question: ¿La revista sale los lunes? NO
4. El maestro vio que los cajones del pupitre del alumno de la escuela estaban llenos.
5. La gente dice que las flores del jardín de la posada de la aldea son preciosas.
6. Todos dicen que las fotos de la portada de la revista del quiosco son muy buenas.

   Question: ¿Son buenas las fotos? YES
7. El conductor vio que la rueda del remolque del camión de las mudanzas estaba pinchada.
8. La gente decía que las letras del cartel de la fachada del almacén eran feas.
9. Ayer supimos que las grietas del cristal del mirador del salón eran muy grandes.
10. El cura anunció que las campanas de la torre de la parroquia del valle estaban viejas.
11. La prensa anunció que las tumbas de la cripta de la catedral de la villa sufrieron daños.
12. El mozo vio que los cojines del sofá del vestíbulo del albergue estaban sucios.

    Question: ¿Estaban limpios los cojines? NO
13. Me han dicho que las llaves del portero de la finca del duque han desaparecido.
14. Hemos visto que los dibujos del cuaderno de la sobrina del pintor eran muy bonitos.
15. La prensa opina que los discursos del líder del partido de la oposición son muy persuasivos.
16. Acabo de ver que la tapa del baúl de los vestidos de la actriz está abierta.

FILLER SENTENCES

1. El mecánico vio que la hélice del motor estaba estropeada
2. El corrector dijo que los párrafos del capítulo eran demasiado largos

   Question: ¿Eran largos los párrafos? YES
3. La modista cree que los botones del abrigo son demasiado grandes
4. La prensa opina que la insignia del equipo está anticuada
5. No nos gusta que las fotos del libro estén en blanco y negro

6. Hemos comprobado que las sillas del estudio son comodísimas
   Question: ¿Las sillas estaban en la cocina? NO
7. Los vecinos dicen que los ladridos del perro no les dejan dormir
8. Me parece que los lazos del vestido son muy vistosos
9. He comprobado que las bombillas de la lámpara están fundidas
   Question: ¿Están fundidas las bombillas? YES
10. El cajero vio que los números de la cuenta estaban equivocados
11. Mi madre ha visto que los marcos de los espejos del tocador de su cuarto están rotos
12. Nos desagrada que la luz de las farolas de las plazas de la ciudad sea amarilla

1.2. Coordinate-NPs: List of experimental and filler sentences used in Subexperiment 1A, including comprehension questions and their correct answers.

EXPERIMENTAL SENTENCES

1. Alguien dijo que los mosquitos, las moscas, las hormigas y la araña estaban muertos.
2. María vio que la mesilla, el armario, la butaca y la estufa estaban rotos.
3. Me pareció que el título, la trama los diálogos y los chistes eran muy graciosos.
4. El maestro vio que los tubos, la vasija, el bidón y la garrafa estaban llenos.
   Question: ¿El bidón estaba lleno? YES
5. La gente dice que la blusa, el abrigo, la chaqueta y la bufanda son preciosos.
6. Todos dicen que el guión, la dirección, los actores y la música son muy buenos.
7. El mecánico vio que el cambio, el embrague, el freno y las llantas estaban estropeados.
   Question: ¿Estaba estropeado el volante? NO
8. La gente decía que las sobrinas, el yerno, la abuela y el suegro eran feos.
9. Los niños creen que las motos, los camiones, los trenes y los aviones son muy grandes.
10. El alcalde anunció que las alfombras, las cortinas los tapices y el telón estaban viejos.
11. La prensa decía que los tejados, las cornisas, las fachadas y los muros sufrieron daños.
12. La camarera vio que las botellas los vasos los platos y los cubiertos estaban sucios.
13. Me han dicho que las joyas los relojes, las tarjetas y las monedas han desaparecido.
14. Hemos visto que los cuadros los carteles, las fotos y los dibujos eran muy bonitos.
    Question: ¿Eran bonitos los carteles? YES
15. Acaban de decirme que los museos los colegios, las tiendas y los bancos están abiertos.
16. La prensa opina que la novela, el documental, el reportaje y la película son muy persuasivos.

FILLER SENTENCES

1. El mecánico vio que la palanca y el muelle estaban estropeados.
2. El corrector dijo que los párrafos y los títulos eran demasiado, largos.
3. La modista cree que los vestidos y el abrigo son demasiado grandes.
4. La prensa opina que la foto y el comentario están anticuados.
5. Mi madre ha visto que los vasos y la cafetera están rotos.
   Question: ¿Están rotos los platos? NO
6. Nos desagrada que la luz y las plantas sean amarillas.
7. Los vecinos dicen que los ruidos y las voces no les dejan dormir.
   Question: ¿Duermen bien los vecinos? YES
8. Me parece que los sombreros y el traje son muy vistosos.
9. He comprobado que la lámpara y la linterna están fundidas.
10. El bailarín notó que los adornos y los flecos se habían desprendido.
    Question: ¿Los adornos y los flecos estaban bien sujetos? NO
11. No nos gusta que los carteles, los anuncios, las fotos y la película estén en blanco y negro.
12. El sastre ha dicho que los tejidos, las telas, el forro y los hilos son muy bastos.

SUBEXPERIMENT 1B

1.3. Embedded-PPs: List of experimental and filler sentences used in Subexperiment 1B, including comprehension questions and their correct answers.

EXPERIMENTAL SENTENCES

1. Alguien dijo que los flachos de las plondas de las busas de la gramera estaban muertos.
2. María vio que las obrijas del gafor de la blesaca de la ironda estaban rotas.
3. Me pareció que las nobesas del brucio de la filoca del petiso eran muy graciosas.
   Question: ¿El petiso era muy gracioso? NO
4. El maestro vio que los gadones del palojo del chibre de la gurtana estaban llenos.
5. La gente dice que las clotas del terín de la rufa del perismo son preciosas.
6. Todos dicen que las bolnas de la garufa de la inofa del buralo son muy buenas.
   Question: ¿Son buenas las bolnas? YES
7. El conductor vio que la mita del frolador del fustio de las gabrinas estaba pinchada.
8. La gente decía que las manigas del opalio de la mátira del gurto eran feas.
9. Ayer vimos que las logas del nordal de la namira del nofón eran muy grandes.
10. El alcalde anunció que las calonas de la tusa de la onubia del plandro estaban viejas.
11. La prensa anunció que las fanelas de la crasta de la mirta de la pasila sufrieron daños.

12. El mozo vio que los algubios del ferno del trafo del lozal estaban sucios.
Question: ¿Estaba sucio el ferno? NO
13. Me han dicho que las afintas del glador de la tustia del fano han desaparecido.
14. Hemos visto que los facucios del atiso de la rodia del bredo eran muy bonitos.
15. Acabo de ver que la chada del fanul de los fonastos del mador está abierta.
16. La prensa opina que los trafonos del greno del púndido de la farnición son muy persuasivos.

FILLER SENTENCES

1. El mecánico vio que la fímula del bosto estaba estropeada.
2. El corrector dijo que los sárulos del canístico eran demasiado largos.
Question: ¿Eran largos los sárulos? YES
3. La modista cree que los falones del namido son demasiado grandes.
4. La prensa opina que la ustania del aropo está anticuada.
5. No nos gusta que las mufas del nucio estén en blanco y negro.
6. Hemos comprobado que las runcas del histolio son comodísimas.
Question: ¿Las runcas eran incómodas? NO
7. Los vecinos dicen que los gulidos del crasto no les dejan dormir.
8. Me parece que los canos del banisto son muy vistosos.
9. He comprobado que las barundas de la sánida están fundidas.
Question: ¿Están fundidas las barundas? YES
10. El cajero vio que los manecos de la turma estaban equivocados.
11. Mi madre ha visto que los murtos del brúnido del ranelor del crambo están rotos.
12. Nos desagrada que la calia de las traconas de las plaudas de las curatas sea amarilla.

1.4. Coordinate-NPs: List of experimental and filler sentences used in Subexperiment 1B, including comprehension questions and their correct answers.

EXPERIMENTAL SENTENCES

1. Alguien dijo que los flachos, las plondas, las busas y la gramera estaban muertos.
2. María vio que las obrijas, el gafor, la blesaca y la ironda estaban rotos.
3. Me pareció que las nobesas, el brucio, la filoca y el petiso eran muy graciosos.
4. El maestro vio que los gadones, el palojo, el chibre y la gurtana estaban llenos.
Question: ¿El palojo estaba lleno? YES
5. La gente dice que las clotas, el terín, la rufa y el perismo son preciosos.
6. Todos dicen que las bolnas, la garufa, la inofa y el buralo son muy buenos.
7. El conductor vio que la mita, el frolador, el fustio y las gabrinas estaban estropeados.

Question: ¿Estaba bien el frolador? NO

8. La gente decía que las manigas, el opalio, la mátira y el glurto eran feos.
9. Ayer vimos que las logas, el nordal, la namira y el nofón eran muy grandes.
10. El alcalde anunció que las calonas, la tusa, la onubia y el plandro estaban viejos.
11. La prensa anunció que las fanelas, la crasta, la mirta y la pasila sufrieron daños.
12. La camarera vio que las algubias, la ferna, el trafo y el lozal estaban sucios.
13. Me han dicho que las afintas, el glador, la tustia y el fano han desaparecido.
14. Hemos visto que los facucios, el atiso, la rodia y el bredo eran muy bonitos.

Question: ¿Era bonito el atiso? YES

15. Acabo ver que la chada, el fanul, los fonastos y el mador están abiertos.
16. La prensa opina que la nucia, el trofano, el retolano y el greno son muy persuasivos.

FILLER SENTENCES

1. El mecánico vio que la fímula y el bosto estaban estropeados.
2. El corrector dijo que los sárulos y el canístico eran demasiado largos.
3. La modista cree que los falones y el namido son demasiado grandes.
4. La prensa opina que la ustania y el aropo están anticuados.
5. Mi madre ha visto que los astajos y el ranelor están rotos.

Question: ¿Los astajos están enteros? NO

6. Nos desagrada que la calia y las traconas sean amarillas.
7. Los vecinos dicen que los gulidos y el crasto no les dejan dormir.

Question: ¿Los gulidos les molestan? YES

8. Me parece que los canos y el banisto son muy vistosos.
9. He comprobado que las barundas y la sánida están fundidas.
10. El bailarín notó que las franas y el bosardo se habían desprendido.

Question: ¿Las franas y el bosardo estaban bien sujetos? NO

11. No nos gusta que los foranes, los camunos, las mufas y el nulio estén en blanco y negro.
12. El sastre ha dicho que los lumidos, las palacas, el findo y los fanes son muy bastos.

APPENDIX 2: LIST OF MATERIALS USED IN EXPERIMENT 2

List of experimental and filler sentences used in Experiment 2.

SUBEXPERIMENT 2A: WORDS

Experimental items

1. Las palabras son: mosquitos, moscas, hormigas, arañas.
2. Escucha con atención: mesilla, armario, butaca, estufa.
3. Tienes que recordar: título, trama, diálogos, chistes.
4. Recuerda las palabras: tubos, vasija, bidón, garrafa.
5. Las palabras son: blusa, abrigo, chaqueta, bufanda.
6. Escucha con atención: guion, dirección, actores, música.
7. Tienes que recordar: cambio, embrague, freno, llantas.
8. Las palabras son: motos, camiones, trenes, aviones.
9. Tienes que recordar: tejados, cornisas, fachadas, muros.
10. Recuerda las palabras: botellas, vasos, platos, cubiertos.
11. Escucha con atención: cuadros, carteles, fotos, dibujos.
12. Tienes que recordar: museos, colegios, tiendas, bancos.

Filler items

1. Recuerda las palabras: sobrinas, yerno, abuela, suegro.
2. Escucha con atención: alfombras, cortinas, tapices, telón.
3. Las palabras son: joyas, relojes, tarjetas, monedas.
4. Recuerda las palabras: novela, documental, reportaje, películas.

SUBEXPERIMENT 2B: PSEUDOWORDS

Experimental items

1. Las pseudopalabras son: flachos, plondas, busa, gramera.
2. Escucha con atención: tobrijas, gafor, blesaca, hironda.
3. Las pseudopalabras son: nobesas, brucio, filoca, petiso.
4. Escucha con atención: gadones, palojo, chibre, gurtana.
5. Las pseudopalabras son: clotas, terín, rufa, perismo.
6. Tienes que recordar: bolnas, garufa, inofa, buralo.
7. Escucha con atención: mita, frolador, fustio, gabrinas.
8. Las pseudopalabras son: logas, nordal, namira, nofón.
9. Escucha con atención: fanelas, crasta, mirta, pasila.
10. Tienes que recordar: salgubias, ferna, trafo, nozal.
11. Escucha con atención: facucios, atiso, rodia, bredo.
12. Tienes que recordar: chada, fanul, fonastos, mador.

Filler items

1. Tienes que recordar: manigas, opalio, mátira, glurto.
2. Tienes que recordar: calonas, tusa, onubia, plandro.
3. Las pseudopalabras son: tafintas, glador, tustie, fano.
4. Escucha con atención: nucia, tofano, retolano, greno.

3. Principales conclusiones

Las conclusiones más importantes que se derivan de este trabajo pueden agruparse en función de los dos tipos de investigación llevados a cabo, a saber, unas conclusiones derivadas de la investigación formal (lógica, gramatical o conceptual) y otras derivadas de la investigación experimental.

Con respecto a las conclusiones derivadas de la investigación gramatical o conceptual, podemos destacar las siguientes[14]:

1. El significado de 'recursión' tiene su origen en la Lógica Matemática y la Teoría de a Computabilidad.
2. Dicho significado –uso– es el que se deriva del Principio de Recursión, que garantiza la buena definición de diferentes funciones. Tal principio constituye la definición por recursión.
3. Los conceptos de recursión y auto-inclusión son independientes. Es decir, ninguno es un subtipo del otro. Ambos conceptos tienen usos –significados– diferentes.
4. Lo concluido en 3 se mantiene aunque pueda decirse que la recursión y la auto-inclusión sean subtipos de auto-referencia. No obstante, la noción de auto-referencia y la noción de recursión *no* son sinónimas, puesto que hay instancias de auto-referencia que no son recursivas (en sentido original).
5. Como también se desprende de este trabajo, las nociones de infinitud discreta y de recursión tampoco coinciden; es decir, no son conceptos coextensivos, dado que se puede hablar de recursión sin infinitud y de infinitud sin recursión. Dicho de otra manera ligeramente distinta, un sistema dotado de infinitud no tiene, necesariamente, que estar caracterizado por la propiedad de la recursión. Y a la inversa, un sistema definido por recursión no tiene que llevar necesariamente a una generación potencialmente infinita

[14] Es importante tener presente que la investigación gramatical, en el sentido en el que, salvo que se indique lo contrario, se usa este adjetivo en el contexto de esta tesis doctoral, está dirigida al uso de los conceptos. Así, la gramática constituye el significado de las palabras y no se puede justificar ni apelando a una supuesta realidad a la cual los significados se refieren ni apelando a si son correctas o adecuadas a los significados, pues sin tales reglas dichas palabras no tendrían significado.

de valores. Así, una función recursiva parcial puede dar un único valor o ninguno. Por otro lado, la infinitud discreta puede caracterizar a un sistema que se implemente iterativamente y que no haga uso de un proceso recursivo en ese nivel

Otras conclusiones sobre la investigación formal, que van más allá de la clarificación conceptual del término 'recursión' son:

1. La controversia entre la dualidad de un algoritmo considerado o bien como un recursor o bien como una máquina de estados abstracta puede resolverse si definimos un algoritmo como una serie de formas y damos la tripla $[\tilde{\sigma}, \tilde{\varepsilon}, \tilde{O}(\tilde{\varepsilon})]$. De este modo, su definición puede ser recursiva, mientras que su implementación puede ser o bien recursiva o bien iterativa (pero esto no es esencial a su definición formal); es decir, ninguna de las dos opciones es esencial con respecto a la otra. Dicho de otro modo, ninguna de las dos opciones prevalece sobre la otra.
2. En mi opinión, las observaciones expresadas en la anterior conclusión sitúan a Wittgenstein en una posición al menos de interés en lo que respecta a los desarrollos de la Lógica Matemática y de la Teoría de la Computabilidad, algo que no siempre es destacado.

En relación con la investigación experimental, voy a considerar dos clases de conclusiones, a saber, aquellas directamente relacionadas con los resultados empíricos y otras relacionadas con la pertinencia de las investigaciones empíricas para explorar el concepto de recursión, que examinaré en el siguiente apartado.

Empezaré, por tanto, exponiendo las principales conclusiones derivadas de la investigación empírica, en concreto, daré cuenta de las siguientes conclusiones generales: (1) sobre cómo se procesan estructuras jerárquicamente complejas frente a secuencias de elementos sin estructura; (2) sobre la irrelevancia del contenido léxico para ciertos procesos relacionados con la gestión de recursos de memoria en el procesamiento; y (3) sobre alcance de la metodología (la tarea) empleada en los experimentos.

Con respecto a (1) conviene recordar que los participantes necesitan más tiempo para detectar el tono distractor, el 'click', incrustado dentro de un sintagma nominal complejo en una tarea de comprensión de oraciones cuando el tono está localizado en posiciones posteriores dentro del sintagma nominal que cuando el tono se localiza en posiciones más tempranas o anteriores; así como que tal hallazgo ha sido totalmente opuesto al que se ha obtenido en relación con las secuencias desestructuradas tanto de palabras como de pseudo-palabras.

Así pues, en relación con la pregunta relacionada con *cómo* se procesan estructuras jerárquicas complejas frente a secuencias desestructuradas, y teniendo en cuenta la tarea realizada en los experimentos, los datos apuntan a que, al menos, se puede hablar de dos estrategias de procesamiento diferentes. En el caso de las estructuras jerárquicas complejas (tanto subordinadas como coordinadas), los participantes parecen recuperar objetos sintácticos previamente procesados cada vez que reciben un objeto sintáctico nuevo, a juzgar por el patrón en los tiempos de reacción. Esto, claro está, aumenta las demandas del procesamiento. Es importante apuntar que esta reactivación no parece estar basada en el cotejo de rasgos morfológicos –los denominados rasgos-*φ*–, como los rasgos de género y número, al objeto de establecer la concordancia del sintagma nominal complejo con su predicado, situado al final de la oración, una operación que supone el establecimiento de relaciones sintácticas de dependencia a larga distancia entre ambos constituyentes. Igualmente importante es apuntar que, quizá, la tarea realizada no sea lo suficientemente sensible a operaciones de esta naturaleza. En todo caso, si el procesamiento hubiera estado basado en tales rasgos y dadas las diferencias en cuanto a las demandas de operaciones de concordancia entre unas estructuras jerárquicas complejas (i.e., las subordinadas) y otras (i.e., las coordinadas), hubiera sido esperable encontrar diferencias. Es más, hubiera sido esperable encontrar diferencias entre los subexperimentos 1A y 1B, pero de esto hablaré en (2). Así las cosas, y en lo que respecta al procesamiento de estructuras jerárquicas complejas (subordinadas y coordinadas, y compuestas tanto de palabras como de psuedo-palabras), los resultados parecen apuntar a un mayor peso de la información sintáctica.

En relación con el procesamiento de secuencias desestructuradas, tanto formadas por palabras (subexperimento 2A) como por pseudo-palabras (subexperimento 2B), la estrategia, como decía, parece haber sido muy diferente, según muestran las diferencias registradas en el patrón de tiempos de reacción entre las estructuras subordinadas y coordinadas (experimento 1), por un lado, y el procesamiento de listas no estructuradas de elementos (experimento 2), por otro. En el caso de las secuencias de palabras/pseudo-palabras, el tiempo de reacción a los tonos distractores situados en las primeras posiciones fue más lento que los situados en posiciones más retrasadas, a la inversa que en las estructuras.

Estos resultados sugieren que las secuencias desprovistas de estructura generan un tipo de procesamiento diferente al que se aplica a las estructuras subordinadas y coordinadas, que, como hemos visto, comparten demandas de procesamiento muy similares. Esto, a su vez, parece indicar que el procesamiento que subyace a la tarea está determinado más por factores asociados a la configuración estructural de los materiales (i.e., su estructura binaria) que a factores relacionados con operaciones de concordancia, siendo esto últimos secundarios a la información puramente 'configuracional'. Por ende, los resultados comparados de los experimentos 1 y 2 parecen mostrar que la sensibilidad de los participantes a los estímulos distractores viene determinada por operaciones concernientes al análisis sintáctico y no por factores puramente perceptivos, como pueda ser la posición serial del estímulo (en el que se presenta el tono distractor) en la serie.

En relación con (2), y tal y como ya he apuntado más arriba, el procesamiento de estructuras jerárquicas complejas no parece estar basado en el contenido léxico de los enunciados o secuencias, o si lo está, nuestra tarea no ha sido sensible a ello. De haber sido así, entonces hubiera sido esperable encontrar diferencias entre los patrones de tiempo hallados en los subexperimentos de cada experimento, dado que en un caso (subexperimentos 1A y 2A) hay información léxica disponible, mientras que en el otro no (subexperimentos 1B y 2B). En cambio, a la vista de los resultados obtenidos, el procesador parece adherirse a las mismas estrategias de procesamiento que emplea con secuencias estructuradas y con

secuencias desestructuradas de estímulos lingüísticos con independencia de su contenido léxico y, a la postre, del significado de los enunciados. Un posible resultado que no ha llegado a producirse, pero que hubiera sugerido una influencia del léxico en los procesos examinados en este estudio, es que hubiera habido diferencias entre oraciones subordinadas y coordinadas tan sólo en el subexperimento 1A (pero no así en el 1B), es decir, en oraciones con palabras y, por tanto, significado. Esto hubiera indicado una contribución de factores semánticos al procesamiento sintáctico de los enunciados objeto de estudio.

Finalmente y en relación con (3), he indicado tanto en (1) como en (2) que la tarea empleada en esta investigación puede no reflejar adecuadamente la sensibilidad de los participantes a información sintáctica de grano más fino, como la involucrada en las operaciones de cotejo de rasgos morfológicos para la concordancia. Sin embargo, y en descargo de la tarea, los resultados han sido muy consistentes en lo tocante al patrón de tiempos de reacción en relación con el procesamiento de estructuras jerárquicas complejas. En todo caso, creo imprescindible ampliar tanto el repertorio de materiales como de tareas para comprobar si la consistencia obtenida en los tiempos de reacción tiene alguna extrapolación más allá de las tareas que contemplan la carga de memoria en el procesamiento.

Discusión general: más allá de esta tesis

En primer lugar y más allá de los resultados obtenidos, es importante tratar la idoneidad de la investigación empírica para explorar el concepto de recursión. Así, es importante tener presente que la investigación del concepto de recursión es gramatical o conceptual. Es decir, su significado –uso– y su relación lógica con otros conceptos relacionados requieren una investigación gramatical. Por ejemplo, si el mecanismo computacional que subyace a la facultad del lenguaje está definido o no por recursión no requiere una investigación empírica. Esta cuestión no trata sobre una propiedad natural del lenguaje. Su investigación es gramatical, y matemática en el sentido de que es un tipo especial de función recursiva.

La recursión no es una entidad susceptible de descubrimiento, sino que disponemos de ella como herramienta. Este aspecto es clave para la investigación empírica, y lo es si se distingue entre lo siguiente: toda investigación empírica destinada a comprobar si se sigue una regla recursiva o de otro tipo presupone el concepto de recursión. Por tanto, ninguna investigación empírica será una prueba para confirmar o desconfirmar una regla recursiva (las reglas no dicen nada, no se pueden aseverar o negar). Tal regla se usa como una norma de representación, no como aquello que ha de ser representado. Se usa para caracterizar, no es lo que tiene que ser caracterizado. Por tanto, los algoritmos –constituidos por reglas gramaticales– requieren una investigación gramatical, mientras que saber si se sigue ésta o aquella regla es una cuestión empírica; y los enunciados que expresan ambos niveles son muy diferentes, gramaticales en el primer caso, expresando así reglas, y empíricos en el segundo, esto es, descripciones susceptibles de verificación o falsación.

Por ello, expresiones como “las bases neuronales de la recursión” o como “la exploración experimental de la recursión” no han de ser entendidas como si la recursión fuera una entidad susceptible de investigación empírica. Así, la primera expresión tiene más que ver, en líneas generales, con las bases neuronales del procesamiento de ciertas estructuras; esto es, con qué regiones del cerebro se activan mientras un participante en un experimento recibe y trabaja con cierta información. La segunda expresión tiene más que ver con el seguimiento de un cierto tipo de reglas y tiene un sentido más amplio que la anterior. La propiedad de la recursión es una propiedad formal, no biológica. Así, aunque seamos capaces de construir formalismos definidos por recursión, eso no significa que tengamos que tener unas bases neuronales más o menos específicas para la recursión.

Por otro lado, cuando se habla de la exploración experimental de la recursión, esto no significa que tengamos que descubrir esa propiedad en los procesos, en otra conducta, o en una región cualquiera de la naturaleza; como si fuera una propiedad natural o como si fuera constitutiva de la naturaleza humana. Significa que caracterizamos un proceso o un comportamiento con esa propiedad, es decir, que caracterizamos un proceso o una conducta como una instancia de una regla recursiva. Esto se hace con

base en unos criterios, como por ejemplo, el tiempo que se tarda en detectar un estímulo distractor, o considerando si se han activado ciertas regiones cerebrales que subyacen, por ejemplo, a la memoria de trabajo. El aspecto relevante, en mi opinión, es que, en cierto modo, encontramos en los patrones de tiempos de reacción o de activación de regiones cerebrales lo que sobre ellos proyectamos en nuestro afán de explicación. ¿Qué quiero decir? Que damos sentido a los datos desde una gramática. Es perfectamente posible que otras personas encuentren otra explicación a estos tiempos de reacción sin recurrir a la noción de recursión. Ahora bien, la gramática desde la que se trata de dar sentido a los datos es, en ambos casos, distinta (lo cual no quiere decir que la noción de recursión tenga significados distintos en cada caso). El énfasis que yo pongo no es sobre la explicación, sino sobre la descripción del uso que se hace de 'recursión' (y otros términos relacionados como 'recursivo') en la investigación experimental. Es decir, no es que el procesamiento sea recursivo *porque* los datos son así, *porque* el patrón de tiempos de reacción es como es. Más bien, nuestro modo de interpretar los datos está constituido por la aplicación de una regla recursiva. Así, 'explorar experimentalmente la recursión' significa 'poner a prueba la aplicabilidad de una herramienta constituida previamente a la investigación empírica'. ¿Significa esto que la investigación empírica es irrelevante en Psicología? Dicho en estos términos generales no tiene sentido responder, depende de los problemas a los que nos enfrentemos, o, más bien, de lo que entendamos como problema.

Por último, este trabajo deja algunas consideraciones abiertas, como no podía ser de otra manera, en relación con todas las disciplinas que, de una u otra manera, pueden hacerse eco de los asuntos examinados en esta tesis, como pueden ser la Biolingüística, la Ciencia Cognitiva, la Filosofía de la Mente, de la Ciencia y del Lenguaje (sin olvidarme de la Filosofía de la Psicología o de la Ciencia Cognitiva, entre otras). Por ejemplo, un aspecto interesante, que puede aunar a todas ellas, es inquirir qué papel juega la recursión en relación con el origen del lenguaje. A tenor de las reflexiones expuestas tanto aquí como en secciones anteriores, entiendo que si la recursión es una propiedad formal aplicada a la definición del mecanismo computacional que subyace a la facultad del lenguaje, no hay nada

susceptible de investigación empírica en ese caso. La razón, vuelvo a insistir, es que la recursión no es una propiedad natural o biológica y, por tanto, no puede estar en la base de la explicación del origen del lenguaje. Antes bien, cualquier explicación en estos términos del origen del lenguaje ya presupone tal noción. Por eso, entre otras cosas, la encontramos dentro de lo que llamamos 'facultad del lenguaje'. Pero esto no es desvirtuar la importancia del trabajo de Chomsky; en mi opinión su trabajo es tremendamente relevante en tanto en cuanto ha construido un procedimiento generativo capaz de dar cuenta de la generación de una cantidad potencialmente infinita de oraciones del lenguaje natural. Ha construido un procedimiento mecánico finito que, quizá, esté a la altura de otras grandes construcciones matemáticas, como la máquina de Turing. Sin embargo, creo que uno de los puntos clave es este: lo ha construido, no lo ha descubierto (como diría Wittgenstein). Es decir, no ha descubierto la naturaleza computacional del lenguaje, sino, más, ha mostrado qué quiere decir 'lenguaje' bajo la perspectiva computacional.

Por último, este trabajo no le quita importancia a la propiedad de la recursión, todo lo contrario. Reconoce su importancia central en el estudio del lenguaje desde una perspectiva computacional, como así ha quedado patente no sólo al mostrar las relaciones y diferencias con respecto a conceptos frecuentemente relacionados, sino también al definir recursivamente el procedimiento mecánico finito chomskiano o al resaltar el papel de la definición por recursión en la definición del concepto forma de algoritmo –esto es, de una serie de formas–, del que el procedimiento mecánico de Chomsky no es sino una instancia. Asimismo, trata de señalar el salto inferencial que hay entre el nivel computacional y el nivel biológico. Salvarlo no consiste en *insertar* en el cerebro el plano computacional esperando que haya un acoplamiento perfecto; esto es, las proposiciones empleadas en un plano computacional no tienen el mismo significado que las empleadas en el nivel neurológico, no son reducibles las unas a las otras. Una investigación matemática, gramatical, no es una investigación biológica; por ello, una investigación matemática o gramatical en relación con el mecanismo computacional no es *ipso facto* una investigación sobre las propiedades biológicas y/o los sustratos neuronales

del lenguaje. Esto, como he tratado de mostrar, está lejos de haberse resuelto satisfactoriamente y, en mi opinión, confunde problemas conceptuales y métodos. Este último aspecto es mucho más amplio de lo que esta tesis puede abarcar, pues sólo ha tratado de reflejar la relevancia del método gramatical para la clarificación de ciertos *problemas*; y con esto quiero llamar la atención sobre "la caracterización de ciertos problemas en psicología como *problemas* y de ciertas prácticas consideradas como su *solución*". Este método, como ya se dijo muchas líneas más arriba, muestra, entre otras cosas, que no todos los *problemas* en Psicología han de resolverse experimentalmente. Y ello no es porque tales problemas queden fuera de la Psicología, al contrario, están en la esencia de la misma, en su gramática.

BIBLIOGRAFÍA

Abelson, H. & Sussman, G. J. with Sussman, J. (1996). *Structure and interpretation of computer programs*. Cambridge, MA: MIT Press.

Abrams, K. & Bever, T. G. (1969). Syntactic structure modifies attention during speech perception and recognition. *The Quarterly Journal of Experimental Psychology*, 21(3), 280-290.

Alemán, A. (2011). *Lógica, matemáticas y realidad*. Madrid: Tecnos.

Arsenijević, B. & Hinzen, W. (2010). Recursion as a human universal and as a primitive. Biolinguistics, 4(2-3), 165-173.

Arsenijević, B. & Hinzen, W. (2012). On the absence of X-within-X recursion in human grammar, *Linguistic Inquiry*, 43, 423-440.

Barwise, K. J. & Moss, L. (1996). *Vicious Circles. On the Mathematics of Non-Wellfounded Phenomena.* Chicago, IL: University of Chicago Press.

Berent, I. & Perfetti, C.A. (1993). An on-line method in studying music parsing. *Cognition*, 46, 203-222.

Berwick, R., Friederici, A.D., Chomsky, N. & Bolhuis, J. (2013). Evolution, brain, and the nature of language, *Trends in Cognitive Science*, *17*, pp. 89-98.

Bever, T. G. & Hurtig, R. R. (1975). Detection of a nonlinguisic stimulus is poorest at the end of a clause. *Journal of Psycholinguistic Research*, 4(1), 1-7.

Blass, A. & Gurevich, Y. (2007). Abstract state machines capture parallel algorithms: correction and extension, *ACM Transactions on Computational Logic*, *V*, 1-29.

Boolos, G. & Jeffrey, R. (1974). *Computability and logic*. Cambridge: Cambridge University Press.

Boolos, G. (1971). The iterative conception of set, *The Journal of Philosophy*, *68*, 215-231.

Brennan, J. & Pylkkänen, L. (2012). The time-course and spatial distribution of brain activity associated with sentence processing. *Neuroimage*, 60, 1139–1148.

Camacho, J. (2003). *The structure of coordination Conjunction and Agreement Phenomena in Spanish and Other Languages*. Dordrecht: Kluwer.

Carroll, J.M. & Tanenhaus, M.K. (1978). Functional clauses and sentence segmentation. *J. of Speech and Hearing Research*, 21, 693-708.

Chomsky, N. & Miller, G. (1963). Introduction to the formal analysis of natural languages. In R.D. Luce, R.R. Bush & E. Galanter (*Eds.*), *Handbook of mathematical psychology, Vol. II* (pp. 269-322). New York: John Wiley.

Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control, 2,* 137-167.

Chomsky, N. (1965). *Aspects of theory of syntax*. Cambridge. MA: MIT Press

Chomsky, N. (1975). *The logical structure of syntactic theory*. New York: Plenum Press.

Chomsky, N. (1980). *Rules and representations*. New York: Columbia University Press.

Chomsky, N. (1995a). *The minimalist program*. Cambridge. MA: MIT Press.

Chomsky, N. (1995b). Language and nature, *Mind, 104*, 1-61.

Chomsky, N. (2005). Three factor in language design, *Linguistic Inquiry, 36,* 1-22.

Chomsky, N. (2006). *Language and mind*. Cambridge: Cambridge University Press.

Chomsky, N. (2007a). Approaching UG from below. In U. Sauerland & H. M. Gärtner (*Eds.*), *Interfaces + Recursion = Language?* (pp. 1-30). Berlin: Mouton.

Chomsky, N. (2007b). Of minds and language, *Biolinguistics, 1,* 9-27.

Chomsky, N. (2008). On phases. In R. Freidin, C. Otero, & M. L. Zubizarreta (*Eds.*), *Foundational issues in linguistic theory* (pp. 133-166). Cambridge. MA: MIT Press.

Chomsky, N. (2010). Some simple evo devo theses: How true might they be for language? In R. Larson, V. Déprez & H. Yamakido (*Eds.*), *The evolution of human language* (pp. 45-62). Cambridge: Cambridge University Press.

Chomsky, N. (2011). Language and other cognitive systems. What is special about language?. *Language Learning and Development, 7*, 263-278.

Chomsky, N. (2012). Some core contested concepts. *Proceedings of the CUNY 2012*, 1-18.

Chomsky, N. (2015). Some core contested concepts, *Journal of Psycholinguistic Research*, *44*, 91-104.

Church, A. (1932). A set of postulates for the foundation of logic, *The Annals of Mathematics*, *33*, 346-366.

Church, A. (1936). An unsolvable problem of elementary number theory. In M. Davis (*Ed.*), *The undecidable* (pp. 88-107). New York: Raven Press.

Cohen, L. & Mehler, J. (1996). Click-monitoring revisited: An on-line study of sentence comprehension. *Memory & Cognition*, 24(1), 94-102.

Corballis, M. C. (2007). Recursion, language and starlings, *Cognitive Science*, *31*, 69-704.

Corballis, M. C. (2011). *The Recursive Mind: The Origins of Human Language, Thought, and Civilization*, Princeton, NJ: Princeton University Press.

Cutland, N. (1980). *Computability: an introduction to recursive function theory*. Cambridge: Cambridge University Press.

Cutler, A., & Norris, D. (1979). Monitoring sentence comprehension. In W.E. Cooper & E.C.T. Walker (eds.), *Sentence processing: Psycholinguistic studies presented to Merrill Garrett* (pp. 113-134). Hillsdale, NJ: Erlbaum.

Davis, M. (1965). *The undecidable*. New York: Raven Press.

Davis, M. (1982). Why Gödel didn't have Church Thesis, *Information and Control*, *54*, 3-24.

Eberhard, K.M., Cutting, J.C., Bock K. (2005). Making sense of syntax: Number agreement in sentence production. *Psychological Review*, 112, 531–559.

Eguren, L. & Fernández Soriano, O. (2004). *Introducción a una sintaxis minimista*. Madrid: Gredos.

Epstein, R. & Carnielli, W. (1989). *Computability: computable functions, logic, and the foundations of mathematics*. Pacific Grove, CA: Wadsworth & Brooks/Cole.

Everett, D. (2005). Cultural constraints on grammar and cognition in Pirahã, *Current Anthropology*, *46*(4), 621-646.

Everett, D. (2009). Pirahã culture and grammar: a response to some criticisms, *Language*, *85*(2), 405-442.

Fernández, G. y Sáez, F. (1987). *Fundamentos de informática*. Madrid: Alianza

Fitch, T. (2010). Three meanings of recursion: key distinctions for biolinguistics. In R. Larson, V. Déprez & H. Yamakido (*Eds*.), *The evolution of human language* (pp. 73-90). Cambridge: Cambridge University Press.

Fitch, W.T., & Hauser, M.D. (2004). Computational constraints on syntactic processing in a nonhuman primate, *Science*, 303, 377-380.

Fitch, W.T., Hauser, M.D. & Chomsky, N. (2005). The evolution of the language faculty: clarifications and implications. *Cognition*,97, 179-210.

Fodor, J. (1992). *A theory of content and other essays*. Cambridge, Mass.: MIT Press.

Fodor, J.A., Bever, T.G. & Garrett, M.F. (1974*)*. *The Psychology of Language: An Introduction to Psycholinguistics and Generative Grammar*. New York: McGraw-Hill.

Forster, K.I. & Forster, J. (2003). DMDX: A Windows display program with millisecond accuracy. *Behavior Research Methods, Instruments, & Computers*, 35(1),116-124.

Frascolla, P. (1994). *Wittgenstein's philosophy of mathematics*. London: Routledge.

Frege, G. (1892). Sobre sentido y referencia. En L. Valdés (*Ed.*), *Ensayos de semántica y filosofía de la lógica* (pp. 84-111). Madrid: Tecnos.

Friederich, R.M. & Friederici, A.D. (2013). Mathematical logic in the human brain: semantics, *PLOS one*, *8*(1): e53699. doi:10.1371/journal.pone.0053699

Friederici, A.D. (2004). Processing local transitions versus long-distance syntactic hierarchies. *Trends in Cognitive Science*, 8, 245-247.

Friederici, A.D., Bahlmann, J., Friedrich, R., & Makuuchi, M. (2011). The neural basis of recursion and complex syntactic hierarchy, *Biolinguistics,* 5, 87-104.

Gentner, T. Q., Fenn, K. M., Margoliash, D., & Nusbaum, H. C. (2006). Recursive syntactic pattern learning by songbirds. *Nature*, 440, 1204–1207.

Gibson, E. (1998). Linguistic complexity: locality of syntactic dependencies. *Cognition,* 68, 1-76.

Gödel, K. (1931). On formally undecidable propositions of the Principia Mathematica and related systems. I. In M. Davis (*Ed.*), *The undecidable* (pp. 4-38). New York: Raven Press.

Gödel, K. (1934). On undecidable propositions of formal mathematical systems. In M. Davis (*Ed.*), *The undecidable* (pp. 39-74). New York: Raven Press.

Gödel, K. (1944). La lógica matemática de Russell, en J. Mosterín (*Ed.*), *Obras Completas* (pp. 313-343). Madrid: Alianza.

Gödel, K. (1964). Postscriptum to Gödel 1931. In M. Davis (*Ed.*), *The undecidable* (pp. 71-73). New York: Raven Press.

Gómez, D.M., Bion, R.A.H., & Mehler, J. (2011). The word segmentation process as revealed by click detection. *Language and Cognitive Processes*, 26(2), 212-223.

Guasch, M., Boada, R., Ferré, P., & Sánchez-Casas, R. (2013). NIM: A Web-based Swiss Army knife to select stimuli for psycholinguistic studies. *Behavior Research Methods, 45, 765-771.*

Gurevich, Y. (2000). Sequential abstract state machines capture sequential algorithms, *ACM Transactions on Computational Logic*, *1*, 77-111.

Gurevich, Y. (2011). What is an algorithm?, *Technical report MSR-TR-2011-116.*

Hakes, D., Evans, J.S., & Brannon, L. (1976). Understanding sentences with relative clauses. *Memory & Cognition*, 4, 283-290.

Hauser, M., Chomsky, N. & Fitch, T. (2002). The faculty of language: What is, who has it, and how did it evolve?, *Science, 298*, 1569-1579.

Holmes, V. M. & Forster, K. I. (1970). Detection of extraneous signals during sentence recognition. *Perception and Psychophysics*, 7(5), 297-301.

Holmes, V.M. (1973). Order of main and subordinate clauses in sentence perception. *J. of Verbal Learning and Verbal Behavior*, 12, 285-293.

Hrbacek, K. & Jech, T. (1999). *Introduction to Set Theory*, New York: Marcel Dekker, Inc.

Hudson, R. A. (1996). The difficulty of (so-called) self-embedded structures. *UCL Working Papers in Linguistics* 8, 283-314.

Humphries, C., Binder, J.R., Medler, D.A., & Liebenthal, E. (2007). Time course of semantic processes during sentence comprehension: an fMRI study. *Neuroimage*, 36(3), 924–932.

Jackendoff, R. & Pinker, S. (2005). The nature of the language faculty and its implications for evolution of language (reply to Fitch, Hauser, and Chomsky), *Cognition*, *97*, 211-225.

Jackendoff, R. (2011). What is the human language faculty? Two views, *Language*, *87*, 586-624.

Johannessen, J.B. (1998). *Coordination.* Oxford, UK: Oxford University Press.

Karlsson, F. (2010a). Syntactic recursion and iteration In H. van der Hulst, (ed.), *Recursion and Human Language*. (pp. 43-67). Berlin/New York: Mouton de Gruyter.

Karlsson, F. (2010b). Multiple final embedding of clauses. *International Journal of Corpus Linguistics*, 15(1), 88-105.

Kenny, A. (1989). *The metaphysics of mind*. Oxford: Oxford University Press.

Kenny, A. (1990). *El legado de Wittgenstein*. Madrid: Siglo XXI.

Kinsella, A. (2010). Was recursion the key step in the evolution of the human language faculty? In H. van der Hulst (*Ed.*), *Recursion and human language* (pp. 179-191). Berlin/New York: Mouton de Gruyter.

Kleene, S. C. (1938). On notation for ordinal numbers, *The Journal of Symbolic Logic*, *3*, 150-5.

Kleene, S. C. (1943). Recursive predicates and quantifiers, *Transactions of the American Mathematical Society*, *53*, 41-73.

Kleene, S. C. (1952). *Introduction to metamathematics*. Amsterdam: North-Holland Publishing.

Kleene, S. C. (2002). *Mathematical logic*. Mineola, NY: Dover Publications.

Lago, S., Shalom, D.E., Sigman, M., Lau, E., & Phillips, C. (2015). Agreement attraction in Spanish comprehension. *Journal of Memory and Language,* 82, 133–149.

Lewis, R.L., Vasishth, S. & Van Dyke, J.A. (2006). Computational principles of working memory in sentence comprehension. *Trends in Cognitive Sciences*, 10 (10), 447-454.

Lobina, D. J. (2011). Recursion and the competence/performance distinction in AGL tasks, *Language and Cognitive Processes*, 26, 1563-1586.

Lobina, D. J. (2014a). Probing recursion. *Cognitive Processing*, DOI 10.1007/s10339-014-0619-z

Lobina, D. J. (2014b). What linguists are talking about when talking about..., *Language Sciences*, 45, 56-70.

Lobina, D.J., Demestre, J., & García-Albea, J.E. (2014). A re-evaluation of the click-detection technique and data. Unpublished manuscript.

Luuk, E. & Luuk, H. (2011). The redundancy of recursion and infinity for natural language, *Cognitive Processing*, *12*, 1-11.

Makuuchi, M., Bahlmann, J., Anwander, A., & Friederici, A.D. (2009). Segregating the core computational faculty of human language from working memory. *PNAS*, 106(20), 8362–8367.

Marion, M. (1995). Wittgenstein and finitism, *Synthese*, *105*, 141-176.

Marion, M. (1998). *Wittgenstein, finitism, and the foundations of mathematics*. Oxford: Oxford University Press.

Marion, M. (2009). Radical anti-realism, Wittgenstein and the length of proofs, *Synthese*, 171, 419-432.

Monk, R. (1990). *Wittgenstein: The duty of genius*. New York: Free Press.

Moro, A. (2008). *The boundaries of Babel*. Cambridge, MA: MIT Press.

Moschovakis, Y. & Paschalis, V. (2008). Elementary algorithms and their implementations. In S.B. Cooper, B. Löwe, & A. Sorbi (*Eds.*), *New computational paradigms* (pp. 87-118). Springer.

Moschovakis, Y. (1998). On founding the theory of algorithms. In H.G. Dales, & G. Oliveri (*Eds*.), *Truth in mathematics* (pp. 71-104). Oxford University Press.

Moschovakis, Y. (2001). What is an algorithm? in B. Engquist & W. Schmid (Eds.), *Mathematics Unlimited: 2001 and Beyond* (pp. 919-936). Berlin: Springer.

Mota, S. (2013). La propiedad de la recursión en el "Tractatus Logico-Philosophicus" de Wittgenstein y su relación con la Teoría de la Computabilidad y la Lógica Matemática, 17, *Observaciones Filosóficas*.
http://www.obervacionesfilosoficas.net/lapropiedaddelarecursion.htm

Mota, S. (2014). La historia y la gramática de la recursión: una precisión desde la obra de Wittgenstein, *Pensamiento y Cultura*, *17*, 20-48.

Mota, S. (2015a). Sobre el concepto de recursión y sus usos, *Praxis Filosófica*, *40*, 153-181.

Mota, S. (2015b). ¿Por qué se usa 'recursión' cuando se quiere significar 'auto-inclusión'?: Clarificaciones conceptuales sobre la recursión en el programa chomskiano, *Revista de Lingüística Teórica y Aplicada*, 53, 171-191.

Mota, S. (2015c). ¿Qué es un algoritmo? Una respuesta desde la obra de Wittgenstein, *Éndoxa. Series Filosóficas*, 36, 317-328.

Mounce, H.O. (1981). *Wittgenstein's Tractatus. An introduction*. Oxford: Blackwell.

Munn, A. (1993). Topics in the syntax and semantics of coordinate structures, Unpublished PhD dissertation, University of Maryland.

Odifreddi, P. (2001). Recursive functions: an archeological look. In C.S. Claude, M.J. Dinneen & S. Sburlan (*Eds*.), *Combinatorics, Computability and Logic* (pp. 13-31). London: Springer-Verlag.

Perruchet, P., & Rey, A. (2005). Does the mastery of center-embedded linguistic structures distinguish humans from nonhuman primates?. *Psychonomic Bulletin & Review*, 12(2), 307-313.

Pinker, S. & Jackendoff, R. (2005). The faculty of language: What's special about it?, *Cognition*, *95*, 201-236.

Pinto, S. (2002). El anti-platonismo del *Tractatus* de Wittgenstein, *Theoria, 13*, 137-152.

Post, E. (1921). Introduction to a general theory of elementary propositions, *American Journal of Mathematics*, *43*, 163-185.

Post, E. (1943). Formal reductions of the general combinatorial decision problem, *American Journal of Mathematics*, *65*, 197-215.

Post, E. (1944). Recursively enumerable sets of positive integers and their decision problems. In M. Davis (*Ed.*), *The undecidable* (pp. 305-337). New York: Raven Press.

Putnam, H. (1972). *Philosophy of logic*. London: George Allen and Unwin.

Pylyshyn, Z.W. (1991). Rules and representations: Chomsky and representational realism. In A. Kashir (*Ed.*), *The Chomskyan turn* (pp. 231-251). Oxford: Basil Blackwell.

Quine, W.V.O. (1980) *From a logical point of view*. Cambridge, Mass. Harvard University Press.

Rey, A., Perruchet, P., & Fagot, J. (2012). Centre-embedded structures are a by-product of associative learning and working memory constrains: Evidence from baboons (*Papio Papio*), *Cognition*, *123*, 180-184.

Roberts, E. (2006). *Thinking Recursively with Java*. Hoboken, NJ: John Wiley and Sons, Inc.

Robinson, R. (1947). Primitive recursive functions, *Bulletin of the American Mathematical Society*, *53*, 925-942.

Rodgers, P., Black, P.E. (2004). Recursive data structure. In: Pieterse, V., Black, P.E. (Eds.), Dictionary of Algorithms and Data Structures. Online at: http://www.nist.gov/dads/HTML/recursivstrc.html.

Rodych, V. (1997). Wittgenstein on mathematical meaningfulness, decidability, and application, *Notre Dame Journal of Formal Logic*, 38, 195-225.

Rodych, V. (1999a). Wittgenstein on irrationals and algorithmic decidability, *Synthese*, 118, 279-304.

Rodych, V. (1999b). Wittgenstein's inversion of Gödel's Theorem, *Erkenntnis*, *51*, 173, 206.

Rodych, V. (2002). Wittgenstein on Gödel: the newly published remarks, *Erkenntnis*, *56*, 379-397.

Rodych, V. (2003). Misunderstanding Gödel: new arguments about Wittgenstein and new remarks by Wittgenstein, *Dialectica*, *57*, 279-313.

Ruiz Abánades, J. (2009). La superación lógica y mística del límite entre yo y mundo: nuevas lecturas del *Tractatus* de Wittgenstein, *Bajo Palabra*, *II Época*, 289-296.

Shanker, S.G. (1987). Wittgenstein versus Turing on nature of Church's thesis, *Notre Dame Journal of Formal Logic*, *28*, 615-649.

Sieg, W. (1997). Step by recursive step: Church´s analysis of effective calculability, *The Bulletin of Symbolic Logic*, *3*, 154-180.

Sieg, W. (2006). Gödel on computability, *Philosphia Mathematica*, *III*, 189-207.

Skolem, T. (1923). The foundations of elementary arithmetic established by means of the recursive mode of thought, without the use of apparent variables ranging over infinite domains. In J. Van Heijenoort (*Ed*.), *From Frege to Gödel. A source book in mathematical logic, 1879-1931* (pp. 302-333). Cambridge, MA: Harvard University Press.

Soare, R. (1996). Computability and recursion, *The Bulletin of Symbolic Logic*, *2*, 284-321.

Soare, R. (1999). The history and concept of computability. In E.R. Griffor (*Ed*.), *Handbook of computability theory* (pp. 3-36,). Amsterdam: North-Holland Publishing.

Soare, R. (2009). Turing oracles machines, online computing, and three displacements in computability theory, *Annals of Pure and Applied Logic*, *160*, 368-399.

Stabler, E.P. (2009). Recursion in gramar and performance. Talk presented at the conference 'Recursion**:** Structural complexity in language and cognition'. Amherst, Massachusetts, may (reprinted in T. Roeper & M. Speas (eds.), Recursion**:** C*omplexity in cognition*, 2014, pp 159-177).

Tomalin, M. (2006). *Linguistics and the Formal Sciences*. Cambridge: Cambridge University Press.

Tomalin, M. (2007). Reconsidering recursion in syntactic theory, *Lingua*, 117, 1784-1800.

Tomalin, M. (2011). Syntactic structures and recursive devices: A legacy of imprecision, Journal of Logic, *Language and Information*, 20, 297-315.

Torretti, R. (1998). *El paraíso de Cantor. La tradición conjuntista en la filosofía matemática*. Santiago de Chile: Editorial Universitaria.

Turing, A. (1936). On computable numbers, with an application to the Entscheidungsproblem. In M. Davis (*Ed.*), *The undecidable* (pp. 116-151). New York: Raven Press.

Van der Hulst, H. (2010). *Recursion and Human Language*, Berlin: Mouton the Gruyter.

Vardi, M. Y. (2012). What is an algorithm? *Communications of the ACM*, *5*(3), 5.

Vicari, G. & Adenzato, M. (2014). Is recursive language-specific? Evidence of recursive mechanisms in the structure of intentional action, *Consciousness and Cognition*, *26*, 169-188.

Vigliocco G, Butterworth B, Semenza C. (1995). Constructing subject–verb agreement in speech: The role of semantic and morphological factors. *Journal of Memory and Language*, 34, 186–215.

Wagers MW, Lau EF, Phillips C. (2009). Agreement attraction in comprehension: Representations and processes. *Journal of Memory and Language*, 61, 206–237.

Wirth, N. (1986). *Algorithms and data structures*. Hemel Hempstead, UK: Prentice Hall. © N. Wirth 1985 (Oberon version: August 2004).

Wittgenstein, L. (1922). *Tractatus logico-philosophicus*. London: Routledge.

Wittgenstein, L. (1958). *Philosophical investigations*. Oxford: Blackwell.

Wittgenstein, L. (1974). *Philosophical grammar*. Oxford: Blackwell.

Wittgenstein, L. (1975a). *Philosophical remarks*. Oxford: Blackwell.

Wittgenstein, L. (1975b). *Wittgenstein's lectures on the foundations of mathematics, Cambridge, 1939*. Chicago: University of Chicago Press.

Wittgenstein, L. (1978). *Remarks on the foundations of mathematics*. Oxford: Blackwell.

Wrigley, M. (1977). Wittgenstein's philosophy of mathematics, *Philosophical Quarterly*, *27*, 50-59.

Yngve, V.H. (1960). A model and a hypothesis for language structure. *Proceedings of the American Philosophical Society,* 104(5), 444-466.

Zhang, N. (2010). *Coordination in Syntax*. Cambridge, UK: Cambridge University Press.

Zwart, J.W. (2011). Recursion in language: A layered-derivation approach, *Biolinguistics, 5*, 43-56.